Sécurité Multicast et Réseaux Ad Hoc

Mohamed Salah Bouassida

Sécurité Multicast et Réseaux Ad Hoc

MÉMOIRE

15 Juin 2003

pour l'obtention du

DEA de l'Université Henri Poincaré – Nancy I

(Spécialité Informatique)

par

Bouassida Mohamed Salah

Composition du jury

	Dominique Méry
	Didier Galmiche
	Noëlle Carbonell
Responsable de filière :	André Shaff
Encadrant :	Isabelle Chrisment

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Mis en page avec la classe thloria.

A mes très chers parents, nulle dédicace n'est susceptible de vous exprimer ma profonde affection, mon immense gratitude pour tous les sacrifices que vous avez consacrés pour moi
A ma petite soeur Marwa
A tous mes amis
A tous ceux qui m'aiment

Remerciements

Je tiens à remercier Monsieur Olivier Festor pour m'avoir donné la possibilité d'effectuer mon stage de DEA au sein de l'équipe de recherche MADYNES.

Je tiens à exprimer ma profonde gratitude et ma sincère reconnaissance à mon encadrante Madame Isabelle Chrisment pour ses directives constructives, sa confiance et toute l'attention dont j'avais été l'objet tout au long de la réalisation de mon stage.

Je remercie vivement mes amis au LORIA pour l'ambiance chaleureuse qu'ils ont pu toujours maintenir, je remercie en particulier Monsieur Abdelkader Lahmadi, Monsieur Adnane Benhalima, Monsieur Adnane Guabtni et Madame Isabelle Astic.

Je tiens à exprimer l'honneur que me font tous les membres du jury en acceptant de juger mon travail. Que tous trouvent ici l'expression de mes sentiments les plus respectueux.

Résumé

Un réseau mobile Ad Hoc est un système autonome de routeurs et de hôtes mobiles connectés par des liaisons sans fil. Ces réseaux offrent la particularité d'être dynamiques dans l'espace et dans le temps. Souvent éphémères, ils ne requièrent aucune infrastructure fixe ; ce qui permet de limiter les coûts de déploiement. Cependant, cette flexibilité associée à la vulnérabilité des liaisons sans fil rend plus prépondérant le besoin de sécuriser les informations des utilisateurs ainsi que celles de signalisation. C'est dans ce cadre que se situe mon stage de DEA, qui consiste à définir les besoins spécifiques en terme de sécurité pour les réseaux Ad Hoc, à évaluer l'impact d'une infrastructure Ad Hoc sur la sécurité multicast et de proposer un modèle de sécurité pour les réseaux Ad Hoc dans un environnement de communications de groupe.

Mots-clés: Ad Hoc, Sécurité, Multicast, Gestion de clés

Abstract

An Ad Hoc network is an autonomous system composed of wireless mobiles nodes. Those networks are dynamic in both space and time, they do not require any fixed infrastructure. However, this flexibility associated with the wireless connections vulnerability, requires more important need of securing users and routing data. Within this context, my DEA intership was to define the security requirements of Ad Hoc networks, to evaluate the impact of an Ad Hoc infrastructure on the multicast security and to propose a security model for Ad Hoc networks within a group communication environment.

Keywords: AdHoc, Security, Multicast, Key management

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Introduction Générale

L'essor des technologies sans fil, offre aujourd'hui de nouvelles perspectives dans le domaine des télécommunications. L'évolution récente des moyens de la communication sans fil a permis la manipulation de l'information à travers des unités de calcul portables qui ont des caractéristiques particulières et accèdent au réseau à travers une interface de communication sans fil. Comparé à l'environnement statique, ce nouvel environnement appelé l'environnement mobile, permet aux unités de calcul une libre mobilité et ne pose aucune restriction sur la localisation des usagers. Les environnements mobiles offrent une grande flexibilité d'emploi. En particulier, ils permettent la mise en réseau des sites dont le câblage serait trop onéreux à réaliser, voire même impossible.

Les réseaux mobiles sans fil peuvent être classés en deux classes : les réseaux avec infrastructure qui utilisent généralement le modèle de la communication cellulaire, et les réseaux sans infrastructure ou les réseaux Ad Hoc. Un réseau Ad Hoc peut être défini comme une collection d'entités mobiles interconnectées par une technologie sans fil formant un réseau temporaire sans l'aide de toute administration ou de tout support fixe. Aucune supposition ou limitation n'étant faite sur la taille du réseau, il est possible que le réseau ait une taille importante.

Comme le rayon de propagation des transmissions des hôtes est limité, et afin que le réseau Ad Hoc reste toujours connecté, il se peut qu'un hôte mobile soit dans l'obligation de demander de l'aide à un autre hôte pour pouvoir communiquer avec son correspondant. La gestion de l'acheminement des données ou le routage, consiste à assurer une stratégie qui garantit, à n'importe quel moment, la connexion entre n'importe quelle paire de nœuds appartenant au réseau. La stratégie de routage doit prendre en considération les changements de la topologie ainsi que les autres caractéristiques du réseau Ad Hoc (bande passante, nombre de liens, ressources du réseau,...). En outre, la méthode adoptée dans le routage, doit offrir le meilleur acheminement des données en respect des différentes métriques de coûts utilisées.

La sécurité constitue actuellement l'un des principaux obstacles à un large déploiement des réseaux Ad Hoc. Sécuriser un réseau Ad Hoc revient à instaurer les différents services de sécurité au sein du réseau, tout en prenant en compte les différentes caractéristiques d'un tel réseau. Parallèlement, le problème de sécurité a fait surface avec l'arrivée des applications multicast dans les réseaux Ad Hoc, comme les conférences virtuelles sur Internet et le télé-enseignement. Sécuriser des communications à destinataires multiples sous le protocole IP, dans les réseaux Ad Hoc, est le cadre dans lequel se situe ce rapport de stage de DEA intitulé "Sécurité Multicast et Réseaux Ad Hoc", réalisé au sein de l'équipe de recherche MADYNES.

Ce document se compose de sept chapitres. Ce chapitre définit les réseaux Ad Hoc et leurs caractéristiques. Le deuxième chapitre présente les services de sécurité pour un réseau Ad Hoc. Dans le troisième chapitre, sont présentés les nouveaux défis qu'apportent les communications multicast en terme de sécurité dans les réseaux Ad Hoc. Ensuite, tout au long du quatrième chapitre, sont présentées quelques approches de gestion de clé dans les groupes multicast pour les réseaux filaires. Le cinquième chapitre est consacré à la proposition d'une nouvelle approche de gestion de clé dans les réseaux Ad Hoc. Des résul-

tats de simulations de la nouvelle approche sont détaillés dans le sixième chapitre. Le septième chapitre conclut le document.

1.2 Définition et Caractéristiques des réseaux Ad Hoc

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calculs portables, poussent aujourd'hui les chercheurs à faire des efforts afin de réaliser le but des réseaux : " L'accès à l'information n'importe où et n'importe quand ". Le concept des réseaux mobiles Ad Hoc essaie d'étendre les notions de la mobilité à toutes les composantes de l'environnement. Aucune administration centralisée n'est disponible, ce sont les hôtes mobiles eux mêmes qui forment d'une manière Ad Hoc une infrastructure du réseau.

Définition : Un réseau mobile Ad Hoc, appelé généralement MANET(Mobile Ad hoc Network), consiste en une grande population, relativement dense, d'unités mobiles qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée. Un réseau Ad Hoc peut être modélisé par un graphe Gt = (Vt, Et), où Vt représente l'ensemble des nœuds et Et modélise l'ensemble des connexions qui existent entre ces nœuds. Si e = (u, v) appartient à Et, cela veut dire que les nœuds u et v sont en mesure de communiquer ensemble directement à l'instant t.

La topologie du réseau peut changer à tout moment, elle est donc dynamique et imprévisible ce qui fait que la déconnexion des unités est très fréquente. La figure 1.1 illustre la dynamicité des réseaux Ad Hoc.

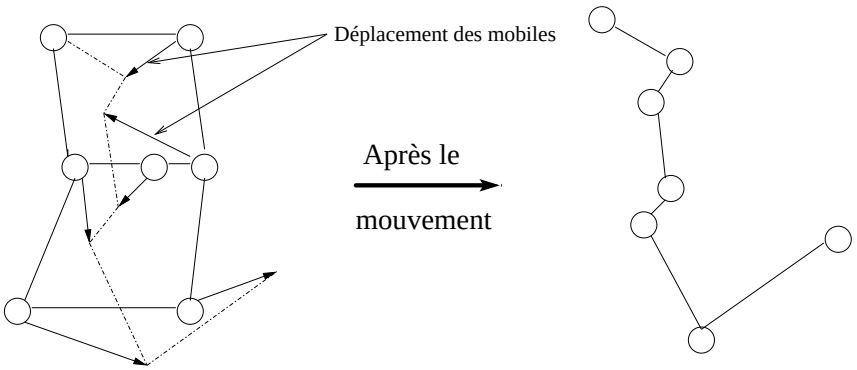


FIG. 1.1 – Changement de la topologie des réseaux Ad Hoc

Applications des réseaux mobiles Ad Hoc

Les applications ayant recours aux réseaux Ad Hoc couvrent un très large spectre, incluant les applications militaires et tactiques, les bases de données parallèles, l'enseignement à distance, les systèmes de fichiers répartis, la simulation distribuée interactive et plus simplement les applications de calcul distribués ou méta-computing.

Caractéristiques des réseaux Ad Hoc

Les réseaux mobiles Ad Hoc sont caractérisés par [T.L00] :

- Une topologie dynamique : les unités mobiles du réseau se déplacent d'une façon libre et arbitraire. Par conséquent, la topologie du réseau peut changer, à des instants imprévisibles, d'une manière rapide et aléatoire. Les liens de la topologie peuvent être unis ou bidirectionnels.
- Une bande passante limitée : une des caractéristiques primordiales des réseaux sans fil est l'utilisation d'un médium de communication partagé ; ce partage fait que la bande passante réservée à un hôte soit modeste.

- Des contraintes d'énergie : les hôtes mobiles sont alimentés par des sources d'énergie autonomes comme les batteries ou les autres sources consommables. Le paramètre d'énergie doit être pris en compte dans tout contrôle fait par le système.
- Une sécurité physique limitée : les réseaux mobiles Ad Hoc sont plus touchés par le paramètre de sécurité, que les réseaux filaires classiques. Cela se justifie par les contraintes et les limitations physiques qui font que le contrôle des données transférées doit être minimisé.
- L'absence d'infrastructure : les réseaux Ad Hoc se distinguent des autres réseaux mobiles par la propriété d'absence d'infrastructure préexistante et de tout genre d'administration centralisée. Les hôtes mobiles sont responsables d'établir et de maintenir la connectivité du réseau de manière continue.

Le chapitre suivant présente les défis en terme de sécurité qu'apportent les réseaux Ad Hoc pour ensuite étudier les différents services de sécurité pour ces réseaux et les différentes méthodes pour les instaurer.

Chapitre 2

Services de Sécurité pour les réseaux Ad Hoc

2.1 Introduction

Dans cette partie, nous identifions les défis de sécurité pour les réseaux Ad Hoc liés à leurs spécificités et caractéristiques, puis nous définissons les services de sécurité qu'on doit instaurer dans un réseau Ad Hoc pour pouvoir faire face aux différentes attaques éventuelles qui peuvent se produire, tout en nous focalisant sur le service d'authentification, qui est la clé de voûte de tout système sécurisé.

2.2 Les défis de sécurité pour un réseau Ad Hoc

- Utiliser des liens sans fils rend un réseau Ad Hoc susceptible d'être exposé à des attaques malicieuses comme des écoutes clandestines, des personnifications actives de communications, ré-envoyer un message ou le déformer. Ces attaques violent ainsi la disponibilité, l'intégrité, l'authentification et la non répudiation. [L.Z99]
- Les réseaux Ad Hoc apportent de nouvelles vulnérabilités comme les attaques par brouillage radio, attaques par consommation de batteries, perturbation du routage Ad Hoc en modifiant les informations de routage. [J.L02]
- Les nœuds d'un réseau Ad Hoc ont une protection physique relativement pauvre, et donc ils ont une probabilité non négligeable d'être compromis. Il s'avère ainsi qu'il ne faut pas considérer seulement les attaques malicieuses venant de l'extérieur du réseau Ad Hoc, mais aussi il faut prendre en compte les attaques venant de l'intérieur par les nœuds compromis. Il est à noter ainsi que l'utilisation d'une entité centrale dans une solution de sécurité pour un réseau Ad Hoc pourrait mener à un état vulnérable du moment où cette entité centrale se trouve compromise.[L.Z99]
- Le service d'authentification dans un réseau Ad Hoc est primordial et s'il est compromis, tous les autres services ne pourront pas être assurés (Attaques sur les mécanismes de sécurité eux mêmes). [J.L02]
- Un réseau Ad Hoc est dynamique à cause des changements fréquents de sa topologie et de ses membres. Ainsi, une solution de sécurité statique pour un réseau Ad Hoc ne suffit pas. Il est indispensable que les mécanismes de sécurité puissent s'adapter rapidement à ces changements. [L.Z99]
- Finalement, un réseau Ad Hoc peut se composer de centaines, voire de milliers de nœuds. Il est important donc que la solution de sécurité proposée permette le passage à l'échelle.

2.3 Authentification

L'authentification permet à un nœud de s'assurer de l'identité des nœuds avec lesquels il communique. Sans authentification, un adversaire peut communiquer avec des nœuds du réseau et ainsi bénéficier de ressources auxquelles il n'est pas autorisé à accéder.

Pour que deux nœuds d'un réseau Ad Hoc puissent communiquer ensemble, la confiance mutuelle se révèle très importante ; en effet, la confiance doit être un prérequis nécéssaire à l'établissement de la communication entre deux nœuds du réseau Ad Hoc pour assurer en conséquence les services de confidentialité, d'authentification et d'autorisation. Afin de garantir l'échange, le service de confiance s'organise selon deux phases distinctes : l'établissement de la confiance, phase initiale pour définir les conditions de l'échange et la gestion de confiance, en assurant son maintien. Pour accomplir l'établissement de la confiance, trois étapes doivent se succéder : la distribution du secret commun, l'établissement d'un canal sûr obtenu grâce au secret commun précédent et l'échange de clés de chiffrement destinées à la confidentialité de la session, si nécessaire.

Le service d'authentification se révèle ainsi indispensable pour pouvoir faire communiquer deux entités appartenant à un réseau Ad Hoc. Dans cette partie, nous allons énumérer différentes approches d'authentification pour un réseau Ad Hoc et voir pour chacune de ces approches, sa manière de traiter la problèmatique de la confiance entre les différents intervenants du réseau.

2.3.1 Key Agreement dans les réseaux Ad Hoc

Le contexte de cette approche est un petit groupe de personnes dans une conférence, présentes ensemble dans une salle pour une réunion Ad Hoc, et faisant partie d'un système asymétrique de cryptage ; ces personnes veulent s'échanger des données secrètement durant la durée de la réunion. Le principe de ce protocole consiste, sachant que tous les nœuds présents ont confiance les uns aux autres, à partager un mot de passe faible à partir duquel un autre mot de passe sera généré et sera la clé de session du groupe.

Ce protocole tel qu'il est présenté par [N.A00] doit avoir les propriétés suivantes :

- Secret : seuls les gens qui connaissent le premier mot de passe doivent être capables de savoir la clé de session résultante.
- Accord contribuant : la clé de session générée doit être composée des contributions des différents participants à la session.
- La tolérance contre les attaques : les attaques prises en compte sont celles qui peuvent insérer des messages mais qui ne peuvent pas modifier ou supprimer des messages envoyés par d'autres personnes.

[N.A00] a commencé par présenter un protocole générique d'authentification EKE (Encrypted Key Exchange) ; les intervenants dans EKE sont deux nœuds A et B dans un réseau Ad Hoc qui veulent se mettre d'accord sur une clé forte de session K, de sorte qu'un attaquant ne puisse pas connaître K et ne puisse non plus attaquer le mot de passe faible de départ p (Dictionary Attack).

Les échanges de messages pour aboutir à la clé de session sont détaillés dans l'annexe.

[N.A00] propose d'étendre le protocole EKE pour qu'il soit un protocole multi-parties, la seule contrainte est qu'un leader doit déclencher les opérations d'authentification et d'échanges de messages ; ce leader constitue ainsi un point de vulnérabilité du réseau Ad Hoc. Ce nouveau protocole tel qu'il a été présenté ne satisfait pas la contrainte d'accord contribuant, car c'est le leader qui calcule la clé de session globale et la diffuse à tous les autres nœuds.

Pour remédier à ce problème, l'auteur a apporté des modifications au protocole de base EKE pour avoir un protocole multi-parties qui permet à tous les intervenants de contribuer à la génération de la clé de session K. Cette modification est très contraignante puisque le leader doit attendre toutes les contributions des autres nœuds pour pouvoir calculer K.

Ce protocole doit être accompagné d'un autre protocole additionnel qui pourrait convaincre les nœuds du réseau que la clé diffusée par le leader est bien la clé correcte du système asymétrique de cryptage. D'autres difficultés dans ce protocole sont détaillées dans [N.A00].

Le protocole de Diffie-Helmann pour l'échange de clés peut être adapté pour effectuer l'authentification via un mot de passe ; ce protocole permet d'éliminer toutes les difficultés rencontrées dans le protocole décrit

précédemment. Ce nouveau protocole fournit un secret partagé parfait entre les différents participants et améliore la tolérance aux pannes.

[N.A00] présente une amélioration de Diffie-Helmann concernant le nombre de messages de communication, en arrangeant tous les participants dans un hypercube.
L'idée de base de ce protocole est illustrée dans la figure 2.1 ; avec quatre participants A, B, C et D, voulant se mettre d'accord sur une clé de session.

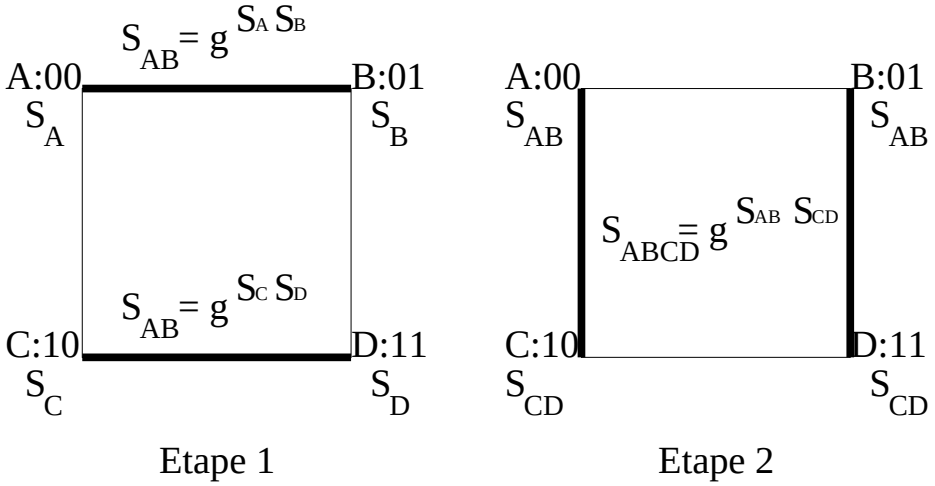


FIG. 2.1 – Echanges Diffie-Helman dans un D-Cube

Chaque participant i a une adresse de deux bits, et génére une contribution S_i. En première étape, les nœuds A et B exécutent Diffie-Helmann pour deux participants, ils parviennent ainsi à calculer $S_{AB} = g^{S_A S_B}$; en même temps, C et D calculent $S_{CD} = g^{S_C S_D}$. La deuxième étape consiste à exécuter Diffie-Helmann entre A et C, et B et D, tout en utilisant comme contributions les clés calculées à l'issue de la première étape. Ainsi, à la fin de la deuxième étape, les quatres participants détiennent la même clé de session $S_{ABCD} = g^{S_{AB} S_{CD}}$.

Maintenant, si on a n=2^d participants, à chaque participant est attribué un sommet dans un hypercube de dimension d. Le protocole procède en d étapes d'échanges de clés avec le même principe présenté ci-dessus. Après les d étapes, tous les participants auront la même clé de session.

Tous ces protocoles présentés résolvent le problème d'authentification sans avoir besoin d'autres infrastructures additionnelles ou de canaux physiques de communication sécurisée, ce qui les rendent adaptés à la nature des réseaux Ad Hoc. Cependant, la dynamicité des réseaux Ad Hoc n'est pas trop prise en compte puisque ces protocoles supposent que la composition de groupe ne change pas durant la session.

2.3.2 Authentification avec cryptographie à seuil

Cette approche présentée par [L.Z99] adopte une infrastructure de clé publique pour bénéficier d'une distribution de clés efficaces et de l'intégrité et de la non répudiation. Chaque nœud du réseau doit avoir une clé publique et une clé privée que le CA (Certification Autority) lui délivre ; le CA, à son tour, a une paire de clés, publique et privée (K,k).

Le CA doit être toujours On Line puisque les clés privées et publiques des nœuds doivent être mises à jour périodiquement pour diminuer le risque d'attaques malicieuses du réseau. Le CA doit aussi remettre en question la clé publique d'un nœud s'il ne lui fait plus confiance. Dans un réseau Ad Hoc, avoir un seul CA pourrait présenter un point de vulnérabilité trop dangereux ; s'il n'est plus disponible, les nœuds ne pourront plus avoir les clés publiques des autres nœuds et ne pourront plus avoir des communications sécurisées entre eux. Les attaquants peuvent aussi utiliser ce point de vulnérabilité pour compromettre le bon fonctionnement de tout le réseau. Une première solution à ce problème consiste à répliquer le CA dans le réseau Ad Hoc, cette solution apporte plus de disponibilité dans le sens où on est sûr qu'au

minimum un CA est On Line à un instant donné, mais cette même solution apporte plus de vulnérabilité au réseau par ce qu'elle réplique aussi la chance de compromettre un CA et par suite compromettre tout le réseau Ad Hoc.

La cryptographie à seuil vient ainsi remédier à ce problème : le nouveau service de gestion de clés ayant la configuration (n,t+1), consiste à avoir n nœuds spéciaux qu'on appelle serveurs, présents dans le réseau Ad Hoc. Chaque serveur a sa propre paire de clés, publique et privée, et enregistre les clés publiques de tous les nœuds du réseau, en particulier, celles des autres serveurs. Ceci permet aux nœuds serveurs d'établir des liens sécurisés entre eux.

[S.Y02] propose de distribuer la confiance à des nœuds qui ont relativement une bonne sécurité physique et une bonne puissance de calcul, surtout dans un environnement hétérogène constitué de nœuds de différentes caractéristiques. L'auteur appelle ces nœuds MOCA (Mobile Certificate Authority).

Dans le cas de notre configuration (n,t+1), les n serveurs partagent la capacité de signer les certificats pour les autres nœuds du réseau. La clé privée de tout le service k est divisée à n secrets partagés (s1, s2, ..., sn), un secret correspond à un serveur. La figure 2.2 illustre cette configuration.

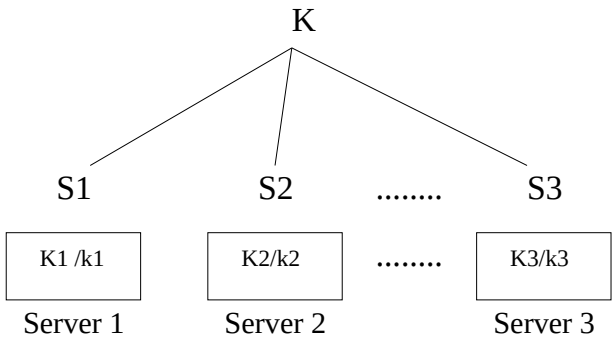


FIG. 2.2 – Configuration du service de gestion de clés

Chaque serveur génère une signature partielle du certificat du nœud et l'envoie à un "combiner", qui a besoin de rassembler au moins t+1 signatures partielles pour générer la signature complète. On suppose que le nombre maximum de serveurs compromis dans n'importe quelle période de temps d'une certaine durée est t ; ainsi, même au cas où on a t serveurs compromis, le "combiner" ne pourra pas générer une signature erronée. [L.Z99] fait l'hypothèse que (n >=3t+1).

Le "combiner" peut aussi vérifier la validité d'une signature partielle envoyée par un serveur, et quand il s'aperçoit qu'une signature est erronée, il la rejette et continue sa collecte de t+1 signatures valides. La figure 2.3 illustre cette opération de construction de signature : Etant donné une configuration (3,2), le serveur 2 a été compromis et le combiner a pu générer la signature de m.

La polémique du choix de t est détaillé dans [S.Y02], le choix d'un t grand est plus sécurisant contre les attaques des adversaires mais en même temps génère plus d'overhead de trafic.

Le "combiner", indispensable pour la génération des signatures des nœuds du réseau, peut à son tour être attaqué et par conséquent devenir un point de vulnérabilité de tout le système de sécurité du réseau. [V.L] propose ainsi une réplication du "combiner" en plusieurs CA : une architecture coopérative où des combiners locaux pourront se former à la volée autour du nœud en question et lui générer sa signature.

[S.Y02] présente un protocole de certification appelé MP (MOCA Certification Protocol). Les clients selon ce protocole envoient des messages Send Request (SREQ) par inondation, chaque MOCA recevant ce message, répond par un message Certif Response (CREP) (protocole analogue au protocole de routage AODV), contenant une signature partielle. Quand le client collecte t+1 CREP valides, il pourra constituer sa signature. Ce protocole n'utilise pas de "combiner", offrant ainsi un meilleur niveau de sécurité.

Concernant l'inondation des SREQ, l'auteur veut éliminer l'inconvénient majeur qui en résulte : tous les MOCAs vont recevoir des copies de SREQ et ils vont tous répondre par des CREP tandis que le client n'a besoin que de t+1 CREP : un overhead trop important. Une solution proposée, est le B-Unicast, qui permet à un nœud d'envoyer en unicast à t+1 MOCAs s'il détient dans sa table de routage assez d'informations concernant leurs routes. Sinon, le nœud se voit obligé de retourner à la solution plus

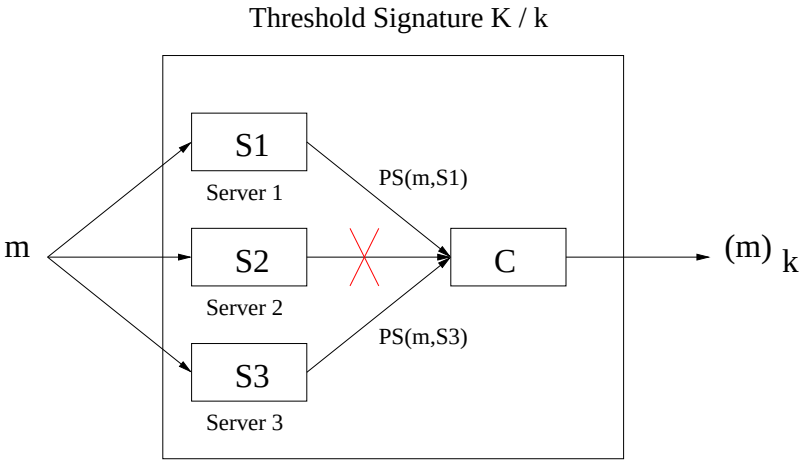


FIG. 2.3 – Cryptographie à Seuil

contraignante d'inondation complète du réseau.

2.3.3 Autres approches d'authentification

[P.U03] cite différentes approches pour la réalisation de l'authentification. Parmi ces mécanismes :

- L'utilisation du SSID (identifiant d'un réseau 802.11 transmis en clair dans les trames) comme un mot de passe lors de la procédure d'association d'une station à un point d'accès (Open Authentication).
- Le filtrage des adresses MAC des stations utilisant les accès sans fil (MAC Address Control List).
- Interdire les services disponibles sur un réseau à un nœud identifié par son adresse MAC, non authentifié. Le client utilise le protocole EAP (Extensible Authentication Protocol), pour être authentifié par son réseau d'accès. Cette architecture d'authentification est applicable en mode filaire et sans fil.

D'une façon générale, Il y a deux choix de base pour l'authentification dans les réseaux Ad Hoc. L'utilisateur a connaissance de ses clés d'authentification, ce choix est appliqué dans les approches d'authentification qu'on a présenté ci-dessus. Le deuxième choix consiste à ce que l'utilisateur ne connaisse pas ses clés d'authentification qui sont la propriété privée du prestataire de service. Une carte à puce par exemple, qui n'est pas duplicable, réalise après délivrance d'un code PIN, les calculs d'authentification.

2.3.4 Conclusion

Différentes approches tentent d'instaurer le service d'authentification dans les réseaux Ad Hoc, ces approches sont caractérisées par une forte ou faible dépendance à un tiers de confiance au sein du réseau. Trois difficultés sont à surmonter pour émuler un tiers de confiance : la répartition sécurisée de la clé privée de l'autorité de confiance mobile, la gestion de l'identité et la génération d'un certificat pour les nœuds mobiles. Ces difficultés sont dues à la forte mobilité, à la compromission potentielle des nœuds et à la dynamicité qui entraîne la perte de références comme l'heure, le nommage et l'adressage. Ces différents problèmes peuvent freiner le déploiement des solutions de confiance surtout dans le cadre de grands réseaux où la mise au point de puissants protocoles cryptographiques d'échange de clés rendrait inutile les tiers de confiance. Cette polémique est plus détaillée dans [V.L].

[V.L] s'intéresse à la localisation et l'exécution des traitements et de leurs données. Les traitements (authentification, contrôle d'accès, autorisation,...) peuvent s'exécuter de manière séparée mais synchronisée et faire appel à des données délocalisées. Cet aspect pose un grave problème de performance c'est pourquoi l'idée de bâtir une architecture unifiée s'avère très fructueuse : regrouper sur un même module les traitements, les politiques de sécurité et les autorisations. Ce module peut être une carte à puce.

Ce module unique permettrait au porteur d'héberger toutes les informations de confiance nécessaires qui doivent accompagner la mobilité de l'usager. Mais s'il est nécessaire de recourir pour la distribution de ces informations à un tiers de confiance émulé, alors la cryptographie à seuil est l'une des solutions appropriées.

2.4 Confidentialité, Non Répudiation et Intégrité

La confidentialité assure qu'une information n'est jamais révélée à une entité non autorisée. Les informations de routage doivent aussi rester confidentielles.

La non répudiation assure que la source d'un message ne peut jamais nier avoir envoyé ce message. Ce service de sécurité est utilisé pour la détection et l'isolation des nœuds compromis dans le réseau.

L'intégrité assure qu'un message transmis dans le réseau n'est jamais corrompu. Un message peut être corrompu à cause d'un affaiblissement dans la propagation Radio ou à cause d'une attaque malicieuse sur le réseau.

Ces trois services de sécurité peuvent résulter d'une architecture de gestion de clés dédiée essentiellement à l'authentification, qui est considérée comme la clé de voûte de la sécurité des environnements sans fil et particulièrement Ad Hoc.

Le protocole WEP (IEEE 802.11) délivre ces trois services qui sont indispensables pour sécuriser un lien Radio.

L'algorithme du protocole WEP (Wired Equivalent Privacy) est détaillé dans [P.U03] ; le successeur de WEP est baptisé TKIP (Temporal Key Integrity), ce protocole utilise une clé maître obtenue au terme de la procédure d'authentification. Une clé de session TK (Temporal Key) est déduite par un algorithme de mise à jour de clés. A partir de cette dernière et d'un paramètre de 48 bits, est déduite une clé de chiffrement de 128 bits utilisée pour un seul paquet. A partir de TK sont calculées aussi des clés réalisant la signature des trames émises et reçues, au moyen d'un code d'authentification de 64 bits appelé MIC (Message Integrity Code). La figure 2.4 illustre ce protocole.

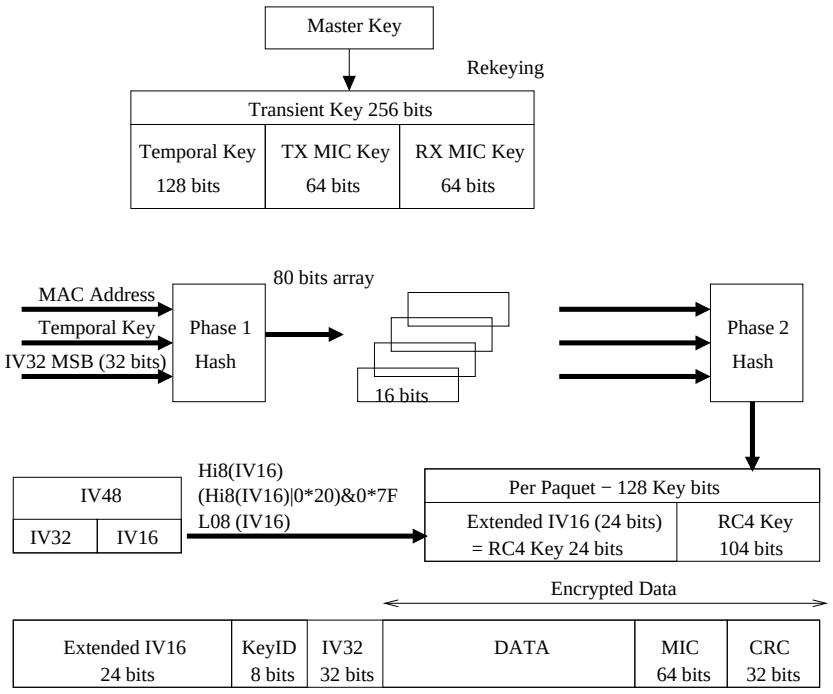


FIG. 2.4 – Le protocole TKIP

2.5 Disponibilité

La disponibilité assure la survie des services fournis par le réseau Ad Hoc malgré les attaques éventuelles de dénis de service. Une attaque de déni de service peut être lancée depuis n'importe quelle couche du réseau Ad Hoc. En effet, dans la couche physique, une attaque pourrait interférer avec les communications ; dans la couche réseau, une attaque peut perturber le protocole de routage et déconnecter le réseau. Les attaques les plus intéressantes selon [F.S99] qui causent un déni de service sont celles qui concernent l'affaiblissement des batteries : un attaquant malicieux peut communiquer avec un nœud dans le seul but est de lui faire consommer l'énergie de sa batterie. Cette attaque s'appelle Sleep Deprivation Torture.

Comme solution à cette attaque, on pourrait restreindre le service aux seuls utilisateurs connus pour éviter les attaques visant la consommation de la batterie. Si un nœud se rend compte qu'il est attaqué, il serait intéressant s'il pouvait stopper les communications avec l'attaquant et en même temps identifier ou repérer la source de l'attaque. Ceci dit, la défense la plus fructueuse contre la Sleep Deprivation Torture serait de déployer un mécanisme de réservation de ressources.

2.6 Autres Problèmes et Conclusion

La nature des réseaux Ad Hoc rend sa sécurisation un problème assez difficile. En effet, tous les services présentés ci-dessus sont indispensables dans une architecture sécurisée de réseaux Ad Hoc, mais demeurent inefficaces contre d'autres attaques des réseaux Ad Hoc, comme le problème de Selfishness et le problème de Wormhole.

Le problème de Selfishness consiste à ce que des nœuds malicieux profitent des avantages et des services du réseau Ad Hoc sans contribuer au bien de tout le réseau, ceci peut être fait en supprimant des paquets de données ou de contrôle et prétendre les router vers d'autres nœuds. Des solutions à ce problème sont détaillées dans [J.H].

Le problème de Wormhole est un cas particulier de Selfishness, il consiste à ce qu'un tunnel ou wormhole est créé dans le réseau entre deux nœuds malicieux reliés via un réseau local. Ce tunnel permettrait à ces nœuds de court-circuiter le flux des messages routés. Cette attaque est particulièrement dangereuse pour les protocoles de routage. En effet, un attaquant pourrait prétendre qu'il est sur le plus court chemin pour aller à une destination donnée, éliminant ainsi toute possibilité de découverte d'autres chemins plus fiables, créant ainsi un déni de service dans le réseau. Des solutions à ce problème sont détaillées dans [Hu02].

Chapitre 3

Sécurité Multicast dans les réseaux Ad Hoc

3.1 Menaces et solutions pour les communications de groupe

Les menaces pour les communications de groupe sont semblables à celles des transmissions point à point. Généralement, elle comprennent l'interception, la génération de trafic non autorisé, la modification et la destruction du trafic multipoint ainsi que l'utilisation illégitime et le déni de service.

Dans le cas du trafic du groupe, le potentiel des attaques est beaucoup plus significatif que celui des transmissions point à point pour les raisons suivantes :[G.C02]

- Le multicast présente plus d'opportunités pour l'interception du trafic ; ce risque d'interception de trafic est beaucoup plus grand pour les réseaux Ad Hoc à cause de la nature de ses liens sans fils.
- Si une attaque se produit, alors un grand nombre de systèmes peuvent être affectés ; dans un environnement Ad Hoc caractérisé par l'absence d'infrastructure fixe, le risque d'affecter d'autres noeuds est beaucoup plus important.
- L'identité et l'adresse du groupe sont connues à large échelle ce qui aide les intrus à diriger leurs attaques.
- Les attaquants peuvent remplacer des membres principaux (membres légitimes du groupe) par d'autres membres illégitimes. Cette attaque prend plus d'ampleur dans le cas des réseaux Ad Hoc, car le risque d'avoir des nœuds internes compromis est plus élevé dans ces réseaux ; ces nœuds pourront ainsi attaquer d'autres entités du réseau et les compromettre. Ce genre d'attaque peut mettre la sécurité de tout le groupe en péril.
- Les informations de routage multicast peuvent aussi être attaquées, interdisant par exemple à une entité de connaître la route exacte à un groupe multicast pour le joindre. Dans les réseaux Ad Hoc, les problèmes de Wormhole et de dénis de service présentés dans le chapitre précédent, accentuent ces risques de vulnérabilité.
- La mobilité et la dynamicité des réseaux Ad Hoc représentent des points de vulnérabilité pour les communications multicast.

Le modèle de communication de groupe peut être sécurisé contre les attaques via l'application de plusieurs services fondamentaux de sécurité tels que le contrôle d'accès, la confidentialité des données, la confidentialité de trafic, l'intégrité, l'authentification de la source et des membres et la non-répudiation de l'émetteur et du récepteur. En perspective de la gestion des clés, seules la confidentialité des données, l'intégrité et l'authentification peuvent être supportées. Les autres services doivent être fournis par les protocoles de communication et dépendent des algorithmes de cryptographie.

Dans la suite de ce chapitre, nous présenterons l'évolution de la vie d'un groupe sécurisé et son impact sur la sécurité des communications du groupe, les différents facteurs pour définir les besoins de sécurité d'un groupe et quelques protocoles de routage multicast pour les réseaux Ad Hoc.

3.2 Evolution de la vie d'un groupe sécurisé

En général, les protocoles multicast présentent deux problèmes qui limitent le passage à l'échelle ou l'extensibilité. Ces problèmes sont 1 affect n et 1 does not equal n.

- 1 affect n : se produit lorsqu'une action chez un membre du groupe affecte tous les membres du groupe.
- 1 does not equal n : apparaît quand un protocole ne peut pas traiter avec tous les membres du groupe; il doit prendre en compte les capacités de chacun.

Les protocoles de gestion de clés multipoints rencontrent le problème de type 1 affect n lors de l'ajout d'un nouveau membre au groupe et les deux types de problèmes lors de la suppression d'un membre :

- Quand un nouveau membre se joint au groupe, l'entité responsable de la gestion de clés doit remplacer la clé du groupe par une autre afin d'empêcher le nouvel abonné d'accéder à l'ancien trafic du groupe. L'ajout d'un seul membre oblige donc tous les autres membres à remplacer la clé de groupe : 1 affect n.
- Quand un membre quitte le groupe, l'entité responsable de la gestion de clés doit remplacer aussi la clé du groupe par une autre afin d'empêcher le membre supprimé d'accéder aux futures communications du groupe. Le gestionnaire de clés est obligé d'utiliser des tunnels sécurisés de communication pour communiquer la nouvelle clé à chaque membre individuellement : 1 affect n car la suppression d'un seul membre oblige les autres membres à remplacer la clé, et 1 does not equal n car le gestionnaire de clés communique la nouvelle clé à un membre comme s'il était indépendant du groupe.

Lors de la mise à jour de la clé de groupe, d'autres problèmes d'extensibilité se manifestent sous forme de trous de sécurité dus à une communication asynchrone entre les différents membres :

- Les membres récepteurs, qui n'arrivent pas à recevoir la nouvelle clé, ne sont plus capables de déchiffrer les communications du groupe. De plus, ils risquent de recevoir des communications envoyées par des membres supprimés.
- Les membres émetteurs, qui n'arrivent pas à recevoir la nouvelle clé, continuent de chiffrer les messages émis avec l'ancienne clé ; les autres membres ne sont plus capables de recevoir les messages de groupe. De plus, des membres supprimés peuvent être capable de déchiffrer les messages ; ce qui compromet la sécurité du groupe.

3.3 Facteurs de sécurité Multicast pour les réseaux Ad Hoc

3.3.1 Type de l'application multicast

Les communications à l'intérieur d'un groupe multicast sont de type one-to-many ou many-to-many. Il existe un spectre d'applications utilisant le multicast IP qui doivent être sécurisés :

Comme exemple, un service d'inscription peut être le one-to-many multicast IP d'une unique source à plusieurs récepteurs. Dans cet exemple, les données transmises doivent être disponibles publiquement ; l'authentification de la source est plus importante que la confidentialité des données.

Un deuxième exemple est un service payant pour une chaîne TV ou un programme à acheter ; bien que les données ne soient pas confidentielles en elles mêmes, le fournisseur de services voudrait bien limiter l'accès à ces données seulement pour les abonnés qui ont payé afin de bénéficier de ce service. Ainsi dans cet exemple, le chiffrement des données est très important pour assurer le contrôle d'accès, surtout dans le cas des réseaux Ad Hoc où le risque des intrusions malicieuses est plus important. L'authentification de la source peut ne pas être importante.

Un troisième exemple serait une conférence qui utilise le many-to-many multicast IP. Dans cet exemple, chaque entité voudrait bien s'assurer de l'identité de la source pour toutes les transmissions dans la conférence. Le chiffrement des données est aussi indispensable pour assurer la confidentialité des données vu qu'il y a des membres bien définis qui peuvent participer à la conférence.

Un autre aspect relié au type de l'application multicast est la fréquence et le taux de la transmission de données. Cet aspect est fortement lié à la performance des algorithmes de cryptage, dans un environnement Ad Hoc où les performances des entités sont limitées. Par exemple, une transmission multicast d'un flux continu vidéo doit avoir un niveau de sécurité différent d'une transmission multicast non fréquente de données ; ceci est dû aux besoins plus importants de calculs des opérations de cryptage pour un flux vidéo.

3.3.2 Taille du groupe et sa dynamicité

La taille d'un réseau Ad Hoc n'est pas contrôlable, le réseau peut avoir une taille importante, et par conséquent un groupe multicast peut avoir un nombre important d'abonnés. La dynamicité des membres qui joignent et quittent le groupe peut aussi être très importante. Ainsi, pour instaurer des mécanismes de sécurité fiables dans les réseaux Ad Hoc, il faut tenir compte de ces deux aspects. La distribution et la densité des membres influent également, à la fois au protocole de routage multicast et au mécanisme de sécurité utilisés.

3.3.3 Issues de passage à l'échelle

La "scalabilité" dans le contexte de sécurité multicast se réfère essentiellement à la capacité des mécanismes de sécurité à s'étendre pour couvrir un groupe de grande taille, sans grande détérioration du niveau des services et des performances du système entier. La scalabilité concerne essentiellement la livraison et la gestion des clés de chiffrement, et aussi la propagation et la gestion des politiques de sécurité.

3.3.4 Modèle de confiance

Quand la cryptographie est employée pour assurer la protection des données, le problème de confiance apparaît. Ce problème engendré concerne les entités qui génèrent, distribuent et gèrent les clés cryptographiques et les politiques de sécurité. Ainsi on a besoin d'un modèle de confiance sous jacent au schéma du modèle de sécurité, qui aborde les problèmes suivants : à quelles entités accorder la confiance pour assurer les services de sécurité requis, quel niveau de confiance faut il leurs accorder, quelle est la source d'autorité,... Ces problèmes prennent plus d'ampleur dans le cas des réseaux Ad Hoc, où le risque d'attaques malicieuses venant de l'intérieur du réseau n'est pas nul, et où le fait d'adopter une seule entité de confiance dans un réseau sans infrastructure, présenterait un point de vulnérabilité trop important et pourrait mettre la sécurité de tout le réseau en péril.

3.4 Les protocoles de routage Multicast dans les réseaux Ad Hoc

3.4.1 Introduction

Afin de pouvoir instaurer une bonne architecture de sécurité multicast dans les réseaux Ad Hoc, il est intéressant d'étudier et de comparer les différents protocoles de routage multicast dans ces réseaux, au-dessus desquels sera mise en œuvre cette architecture.

Il y a deux familles de protocole de routage multicast pour les réseaux Ad Hoc, les protocoles réactifs et les protocoles proactifs.

L'approche réactive ou à la demande conserve la bande passante et réduit l'overhead par l'utilisation des messages de contrôle seulement si nécessaire. Ceci rend ces protocoles plus adaptés aux réseaux denses, très mobiles et avec beaucoup d'activités. Les protocoles proactifs, quant à eux, changent constamment les informations de routage à chaque changement de topologie. Cette approche répare les failles des liens très rapidement et la probabilité de perte de paquets est pratiquement nulle. Ceci rend les protocole proactifs mieux efficaces dans les réseaux peu denses, peu mobiles et avec peu d'activités.

Beaucoup d'algorithmes de routage multicast pour les réseaux Ad Hoc ont été mis au point. Nous allons présenter dans cette section les plus importants : MOLSR [A.L01] qui est un protocole de routage

proactif, MAODV [E.R99], ODMRP [Lee99] et ADMR [J.J01] qui sont des protocoles de routage réactifs, et MZR [V.D00] qui est un protocole hybride.

3.4.2 Multicast Optimised Link State Routing (MOLSR)

Le protocole Multicast Optimised Link State Routing appartient à la famille des protocoles de routage multicast qui maintiennent un arbre de diffusion par source dans un groupe de diffusion. MOLSR est l'extension multicast pour le protocole de routage unicast OLSR. Il est conçu pour tirer profit de ce qui est fait dans le protocole OLSR. Il utilise les différentes tables de routage, de voisinage et de topologie afin de minimiser le trafic de contrôle et d'économiser la bande passante en utilisant les relais multipoints RMP définis dans OLSR. La figure 3.1 montre comment se fait l'inondation par les RMP dans OLSR.

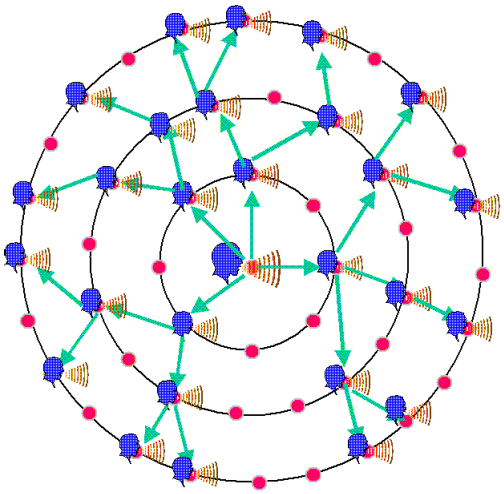


FIG. 3.1 – Inondation par les RMP dans OLSR

Les optimisations apportées par rapport au Link State sont essentiellement les suivantes :

- Un paquet de contrôle d'un nœud ne contient qu'une liste de voisins RMP au lieu de tous les voisins.
- La diffusion des paquets de contrôle ne passe que par les RMP au lieu de l'inondation complète.
- Un RMP me retransmet que s'il reçoit le paquet en premier par le nœud dont il est RMP.
- Les nœuds connaissent une topologie partielle constituée des voisins directs et des RMP des autres nœuds.
- Les routes calculées sont optimales par rapport à la topologie complète.

MOLSR peut fonctionner dans un environnement hétérogène composé de mobiles supportant MOLSR et d'autres qui ne font tourner que le protocole de base OLSR. La seule condition nécessaire est que les mobiles offrant les services multicast assurent une connectivité minimale entre les clients membres du groupe multicast et les sources. Ceci est possible du fait que le trafic de contrôle généré par les mobiles MOLSR sera relayé par tous les mobiles même s'ils ne supportent pas l'extension multicast (caractéristique importante d'OLSR).

Le principe de MOLSR consiste à construire et maintenir un arbre multicast pour chaque tuple (source, groupe) d'une façon distribuée sans aucune entité centrale tout en offrant les plus courts chemins de la source aux membres du groupe. OLSR réagit à chaque modification dans le voisinage ou la topologie en recalculant l'ensemble des relais multipoints et la table de routage. MOLSR profite de ces mises à jour pour optimiser les arbres multicast et détruire les branches obsolètes.

Les mobiles qui n'offrent pas les services multicast peuvent s'abonner à un groupe par l'intermédiaire du protocole WIGMP (Wireless Internet Group Membership Protocol) [A.L03]. Ce protocole, similaire à IGMP, a été adapté à un environnement mobile sans fil. Il permet à un routeur multicast de connaître et gérer les hôtes dans son voisinage.

Construction de l'arbre multicast

Les routeurs multicast se déclarent dans tout le réseau par inondation d'un message MC_CLAIM (utilisant la technique optimisée d'inondation d'OLSR). Ces nœuds ré-envoient ces informations périodiquement. Quand une source veut envoyer des données pour un groupe multicast G, elle envoie un message SOURCE_CLAIM, qui permet aux membres de ce groupe de détecter sa présence et de se joindre à l'arbre multicast associé au groupe.

Quand un membre du groupe reçoit un message SOURCE_CLAIM d'une source et s'il n'est pas encore un participant au tuple (source, groupe), il doit suivre les étapes suivantes : il consulte la table de routage multicast pour avoir le Next-Hop pour joindre la source. Ce nœud Next-Hop sera son parent dans l'arbre multicast du groupe, il envoie un CONFIRM_PARENT à son nœud parent et le nœud parent joint à son tour le tuple (source, groupe), s'il n'est pas participant à cet arbre. Le message CONFIRM_PARENT est remonté Hop by Hop, via les routeurs multicast intermédiaires, ainsi la branche de l'arbre correspondant au nouveau membre sera construite. La figure 3.2 illustre ces différentes étapes.

Maintenance de l'arbre multicast

Les arbres multicast sont périodiquement rafraîchis à chaque envoi de message SOURCE_CLAIM ou CONFIRM_PARENT. Les changements de la topologie sont détectés par les messages de contrôle de OLSR. Les mises à jour de l'arbre multicast sont déclenchées par les changements de la topologie du réseau.

Détachement de l'arbre multicast

Quand un nœud veut quitter un arbre multicast et s'il est une feuille, il se détache de l'arbre et envoie un message LEAVE à son nœud parent. Si ce parent devient une feuille et s'il n'est pas un membre du groupe, il se détache à son tour de l'arbre. Les branches non utilisées de l'arbre sont détruites automatiquement.

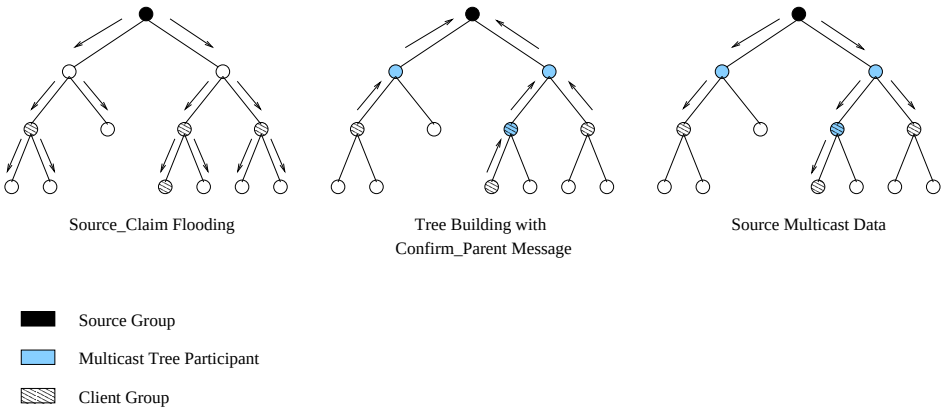


FIG. 3.2 – Construction de l'arbre MOLSR

Le protocole WIGMP et son interaction avec MOLSR

IGMP (Internet Group Management Protocol) est utilisé par les routeurs multicast pour la découverte des membres des groupes dans le réseau. Cependant, dans le contexte des réseaux Ad Hoc, ce protocole présente un inconvénient qui est sa procédure d'élimination des Queries. [A.L03] illustre les limites de IGMP pour les réseaux Ad Hoc avec un exemple schématisé par la figure 3.3.

Le routeur Querier est élu dans IGMP s'il a l'adresse IP la plus petite. Lorsque R2 reçoit un message Querier de la part du routeur R1 qui a l'adresse IP la plus petite, R2 passe à l'état Non Querier. Le problème est que les messages Unsollicited Report envoyés par H ne seront pas reçus par R2.

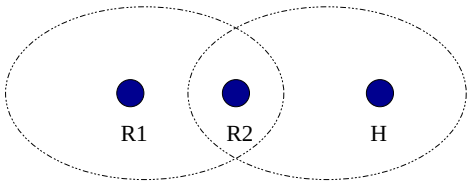


FIG. 3.3 – Limite de IGMP pour les réseaux Ad Hoc

Pour remédier à ce problème, des modifications ont été apportées au comportement des routeurs dans IGMP, ce nouveau protocole est baptisé WIGMP (Wireless IGMP). WIGMP permet ainsi aux routeurs multicast de savoir s'il y a au moins un membre du groupe multicast dans leurs voisinages et s'ils doivent router les données multicast à des nœuds de leurs voisinages.

WIGMP coexiste avec un protocole de routage multicast comme MOLSR, qui est chargé de la construction et de la maintenance de l'arbre multicast et qui permet à tout nœud du réseau Ad Hoc de recevoir des données multicast. Les procédures d'élimination des routeurs Non Queriers et de la désignation du routeur qui envoie les données multicast pour un groupe sont détaillées par [A.L03].

Lorsque un membre quitte un groupe multicast ou lorsque un nœud envoie un Unsollicited Report pour joindre un groupe, WIGMP notifie le protocole de routage multicast MOLSR.

3.4.3 Multicast Ad Hoc On Demand Distance Vector (MAODV)

MAODV est une extension du protocole de routage unicast AODV qui est un protocole réactif. Tout comme MOLSR, MAODV réduit l'overhead par incorporation des fonctionnalités Multicast et Unicast. MAODV utilise le concept de groupe leader qui est similaire au point de rendez vous utilisé dans PIM. Quand un nœud veut joindre ou envoyer à un groupe multicast et n'a pas une route valide pour ce groupe, il diffuse un Route Request Message (RREQ). Si le nœud connaît le groupe leader, il lui envoie en unicast le RREQ. Si un nœud reçoit un join RREQ et s'il est membre du groupe multicast ou s'il connaît une route valide à ce groupe, il répond avec un Route Reply Message (RREP) ; sinon, il ré-innonde le RREQ. La figure 3.4 montre comment se fait la découverte de chemins pour le protocole MAODV.

Si le nœud, source du RREQ, ne reçoit pas un RREP pendant un certain intervalle de temps, il diffuse encore une fois le RREQ, et s'il ne reçoit encore pas de RREP après un nombre d'essais RREQ RETRIES, il fait l'hypothèse qu'il n'y a pas de membres dans ce groupe et devient ainsi le groupe leader.

Un nœud peut recevoir plusieurs RREP suite à la diffusion de RREQ, il doit ainsi choisir la meilleure route en terme de nombre de sauts par rapport à la source et envoie en unicast un message Multicast Activation (MACT) au nœud approprié. Si le récepteur de MACT n'est pas un membre du groupe multicast, il cherche la meilleure route et envoie un MACT à son nœud parent. Ce scénario continue jusqu'à ce que l'émetteur de RREP reçoit un message MACT.

Quand un membre veut quitter un groupe multicast, il envoie un Prune Message à son nœud parent, qui à son tour envoie un autre Prune Message jusqu'à atteindre un membre de groupe.

Malgré la simplicité du broadcast pour la découverte des routes, ce mécanisme présente des inconvénients selon [L.C03]comme la redondance et les collisions. L'auteur présente ainsi une solution qui est l'inondation contrôlée (CF), et l'applique sur AODV. Cette technique a pour but la redécouverte et la réparation de routes, ceci requiert donc une connaissance au préalable d'une route de la source vers la destination. Pour la redécouverte des routes, le message RREQ est dirigé vers la localisation probable de la destination (si la destination a bougé, ses nœuds voisins garderont une piste sur sa nouvelle localisation). La technique d'inondation contrôlée tend ainsi à réduire le trafic de contrôle.

3.4.4 Multicast Zone routing (MZR)

Multicast Zone Routing est une extension du protocole de routage unicast ZRP, qui constitue une approche hybride entre les protocoles de routage proactifs et ceux réactifs. En effet, le réseau ZRP est

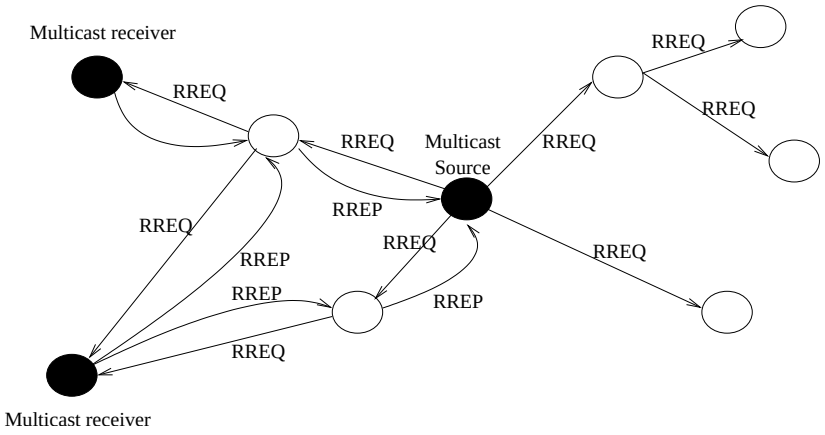


FIG. 3.4 – MAODV Path Discovery

partitionné en zones, le routage au sein d'une zone est proactif, tandis que le routage inter-zones est réactif.

Pour créer une zone, un nœud MZR diffuse un Advertisement Message avec un Time To Live (TTL) égal au rayon de la zone. Les nœuds qui sont à TTL nombres de sauts deviennent des Borders nœuds jouant le rôle de routeurs entre les nœuds à l'intérieur de la zone et le reste du réseau MZR. La figure 3.5 montre un exemple de réseau MZR.

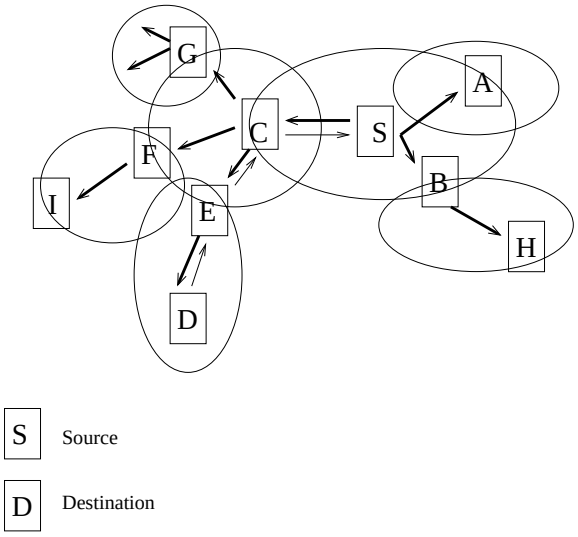


FIG. 3.5 – Exemple de réseau MZR

MZR est un algorithme Source spécifique, c'est à dire, à chaque couple (source, groupe) est associé un arbre multicast. Quand la source veut initier la construction d'un arbre multicast relatif à un groupe, elle envoie un message Tree Create à tous les nœuds dans sa zone. Les nœuds qui reçoivent ce message et qui voudront recevoir des données au sein du groupe multicast, répondent par un Tree Create Ack. Une fois la source termine la construction de l'arbre partiel au sein de sa zone, elle envoie un message Tree Propagate aux nœuds de bord pour étendre la recherche de membres de groupe appartenant à d'autres zones du réseau. Les nœuds de bord, à leurs tours, entament une construction d'arbres multicast dans leurs zones respectives ; cette séquence se répète jusqu'à ce que tous les nœuds du réseau reçoivent un Tree Create de la part de la source du groupe.

[V.D00] suggère d'éliminer le message Tree Propagate et de le remplacer par le fait que les nœuds de bord doivent entamer la construction de l'arbre multicast en dehors de la zone de la source, dès leurs réceptions du message Tree Create.

Quand un nœud veut joindre un groupe multicast ou détecte une faille dans le lien avec son nœud parent, il essaie de construire une branche vers le groupe en émettant un Join à tous les nœuds de sa zone. Si un nœud recevant le Join, connaît une route valide vers le groupe, il lui répond par un Join Ack, sinon le nœud propage sa recherche vers l'extérieur de sa zone par l'envoi du message Join aux nœuds de bord. La même approche se répète jusqu'à ce que le nœud puisse joindre l'arbre multicast.

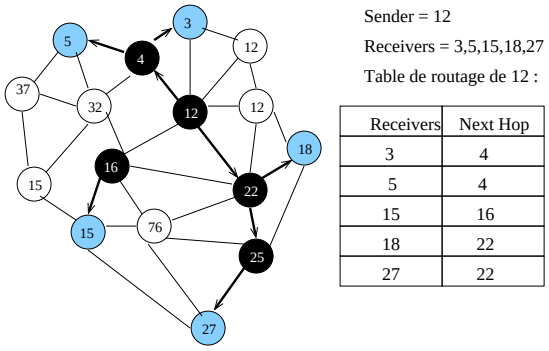
Quitter un groupe multicast se fait par l'envoi d'un message Prune au nœud parent, celui ci décide ou bien de stopper la transmission des données multicast au nœud quittant le groupe, ou bien d'émettre un Prune à son nœud parent pour quitter lui aussi le groupe. Cette approche continue jusqu'à ce que le message Prune parvient à un nœud membre du groupe ou à la source du groupe.

3.4.5 On Demand Multicast Routing Protocol (ODMRP)

ODMRP est un protocole à la demande ; les nœuds maintiennent donc les informations de routage multicast seulement lorsque la session multicast est active. Le groupe multicast correspond à un maillage ou mesh de nœuds dans lequel les paquets multicast sont inondés. Une source qui veut joindre le groupe diffuse le paquet JOIN_QUERY par inondation. Lors de la première réception de ce paquet, un nœud actualise sa table de routage, et re-diffuse le JOIN_QUERY. Si un nœud récepteur de JOIN_QUERY est membre du groupe, il inonde ses voisins avec un paquet JOIN_REPLY contenant une liste des sources multicast et leurs "next hop".

Lors de la réception d'un JOIN_REPLY, un nœud qui se retrouve dans la liste des "next hop", se rend compte qu'il joue le rôle de routeur pour arriver à la source et doit ainsi envoyer à tous ses voisins un nouveau message JOIN_REPLY. De cette façon, l'arbre multicast se crée. La figure 3.6 illustre ce mécanisme par un exemple de tables de routages ODMRP.

Le rafraîchissement de l'arbre multicast se fait par un broadcast périodique de JOIN_REPLY. Ceci permet à un nœud quittant l'arbre multicast de ne pas notifier le groupe explicitement.



Receivers	Next Hop
3	4
5	4
15	16
18	22
27	22

FIG. 3.6 – Exemple de tables de routage ODMRP

3.4.6 Adaptive Demand Driven Multicast Routing (ADMR)

Tout comme MAODV, ADMR est un protocole de routage multicast, à la demande et basé sur la source. Et comme ODMRP, ADMR est indépendant du protocole de routage unicast sous jacent. ADMR permet aux nœuds du réseau de joindre, quitter et envoyer des données à un couple (Source, Groupe) pendant une session multicast. Un nœud qui veut envoyer du trafic multicast utilisant ADMR, commence par inonder le réseau avec le premier paquet de données multicast. Un nœud qui veut joindre le groupe répond en envoyant un paquet Receiver Join. En utilisant les routes parcourues par ces deux types de messages, les nœuds du réseau sont capables de créer les routes multicast.

Au cas où la source n'a plus de données multicast à envoyer, elle diffuse un Keep Alive à l'arbre multicast ; un nœud de l'arbre qui ne reçoit ni données ni message Keep Alive pendant un intervalle de temps donné, suppose donc que le trafic multicast est fini et il aura donc à quitter le groupe.

Au lieu de notifier l'arbre multicast pour quitter le groupe, les nœuds ADMR utilisent les connaissances passives (Passive Acknowledgements) pour déterminer s'ils peuvent quitter le groupe. Cette technique consiste à ce qu'un nœud, juste après avoir routé du trafic multicast à ses nœuds descendants, écoute s'il y en a au moins un qui reçoit ces données ; si ce nœud se rend compte qu'il n'a plus de descendants intéressés par le trafic multicast, il peut quitter le groupe et stopper la transmission du trafic.

3.4.7 Conclusion

Beaucoup de travaux de recherche ont mis au point des protocoles de routage multicast pour les réseaux Ad Hoc, nous avons choisi de citer dans ce document les plus importants.

Nous avons commencé par le protocole MOLSR qui est un protocole proactif, conçu pour les routeurs mobiles et présentant l'avantage de supporter aussi un environnement hétérogène composé de simples routeurs point-à-point OLSR, des routeurs MOLSR et de machines sans protocole de routage. L'inondation selon MOLSR se fait via des multipoints relais ; cette technique tend essentiellement à réduire l'overhead de l'inondation complète du réseau.

Le protocole MZR tend aussi à diminuer l'overhead de l'inondation par partitionnement du réseau en zones, ce protocole peut s'adapter aux réseaux NTDR présenté dans [V.V02]. Dans ces réseaux, il y a un ensemble de clusters, contenant chacun un "clusterhead" ou un chef de cluster. Les différents clusterheads se relient entre eux pour former le backbone de routage. Chaque cluster étant constitué de nœuds à un seul niveau du clusterhead, et les communications inter-clusters sont restreintes aux seuls "clusterheads".

Avec différents degrés de succès, les différents algorithmes de routage multicast essaient d'augmenter la réactivité du protocole et de réduire l'overhead. [Chr02] présente un nouveau protocole de routage multicast qui répond à ces deux critères, ce protocole s'appelle XMMAN (Extensions for Multicast in Mobile Ad Hoc Networks). Une comparaison entre les différentes approches est aussi détaillée dans [Chr02].

Chapitre 4

Différentes approches de gestion de clés de groupe Multicast

4.1 Introduction

La sécurité d'un groupe multicast requiert que seuls les membres du groupe puissent accéder aux données émises par la source, même si ces données sont diffusées dans tout le réseau. Pour assurer cette confidentialité des données, une clé symétrique est utilisée par la source pour crypter les données et par les membres pour les décrypter. Cette clé est appelée TEK (Traffic Encryption Key). La vie d'une session d'un groupe sécurisé est schématisée par un ensemble d'intervalles de temps ; chaque intervalle est défini par un changement de l'état du groupe, c'est à dire une arrivée ou un départ d'une entité membre du groupe. (cf figure 4.1)

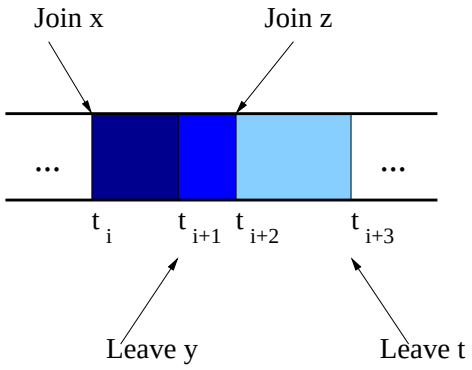


FIG. 4.1 – Evolution de la vie d'un groupe sécurisé

Afin de préserver la sécurité du groupe, il serait indispensable donc, de renouveler la clé du groupe après chaque arrivée ou un départ d'un membre du groupe. En effet, un membre ayant quitté le groupe, ne doit plus être capable de décrypter le trafic après son départ. De même, un nouvel utilisateur, membre du groupe, ne doit pas accéder au trafic envoyé avant son arrivée.
Ainsi, après chaque changement de l'état du groupe, une nouvelle TEK est générée et distribuée à tous les membres du groupe y compris le nouveau en cas d'arrivée, et les membres restants en cas de départ.

Plusieurs architectures de gestion de clés dans les réseaux filaires ont été proposées et élaborées, on peut les classifier en deux approches.

Approche A

Une première approche consiste à partager une seule clé de groupe, utilisée par la source pour crypter les données multicast et par les membres pour les décrypter ; cette approche est utilisée au sein d'une architecture centralisée où un seul serveur est responsable de la génération, du renouvellement et de la distribution de la clé du groupe. Cette approche présente un inconvénient majeur, c'est qu'elle ne permet pas la "scalabilité" du moment où le nombre de messages requis pour renouveler la clé du groupe est égal au nombre des membres du groupe (problème 1 affects n). En plus, l'utilisation d'un seul serveur aboutit à un goulot d'étranglement lors de la phase de renouvellement de clé du groupe, et présente un point de vulnérabilité pour les attaques malicieuses.

Approche B

Une deuxième approche consiste à subdiviser le groupe multicast en plusieurs sous groupes. Chaque sous groupe partage une TEK locale gérée par un contrôleur local. Cette approche se veut hiérarchique pour remédier au problème 1 affects n, rencontré lors de la première approche. En effet, lors d'une arrivée ou un départ dans le groupe, seul le sous groupe concerné par ce changement changera sa TEK locale. Ainsi, cette approche est plus "scalable", mais en contre partie, elle présente l'inconvénient de requérir des opérations de cryptage/décryptage lors de passage d'un sous groupe à un autre.
Un inconvénient commun à ces deux approches est qu'elles ne sont pas flexibles par rapport à la dynamicité du groupe. Ainsi, pour un groupe avec peu de changements de membres, la première approche serait la mieux adaptée ; et pour un groupe qui présente un changement fréquent d'état, la deuxième approche serait la mieux adaptée à cause de son atténuation du phénomène 1 affects n et de son scalabilité.

Dans ce chapitre, nous allons présenter le protocole BAAL [G.C02] qui fait partie de la première approche, le protocole IOLUS [S.M] et le protocole AMAM [H.S03] qui appartiennent à la deuxième approche, et nous terminons par la description d'un protocole hybride baptisé AKMP (An Adaptative Key Management Protocol for Secure Multicast) [H.B02].

4.2 Le protocole BAAL

BAAL est un protocole de gestion de clé de groupe sécurisé, qui a été élaboré au sein de l'équipe de recherche RESEDAS/MADYNES, et assure le contrôle d'accès au groupe, la confidentialité des données et l'authentification des membres du groupe. Ces services sont assurés en partageant une seule clé de groupe K_{grp} ; ce qui rend BAAL un protocole appartenant à la première approche présenté ci-dessus.

Les opérations définies pour la gestion de la clé de groupe sont :

1. Initialisation du groupe : distribution de la clé du groupe à des participants connus à l'avance.
2. Ajout d'une nouvelle entité : renouvellement de K_{grp} après l'ajout de nouvelles entités au groupe.
3. Retrait d'une entité : renouvellement de K_{grp} après le départ d'un membre de groupe.
4. Renouvellement périodique : renouvellement périodique de K_{grp}.

Les acteurs de l'architecture de BAAL sont un contrôleur global, des contrôleurs locaux et des membres de groupe. La figure 4.2 illustre cette architecture.

- Contrôleur Global (CG) : peut être un organisateur de conférences et a le droit de créer un groupe sécurisé sur Internet. Il détient une liste Participant_List, des futurs participants aux communications de groupe ; cette liste est créée à partir d'autres moyens (e.g. e-mail, courrier, fax,...). Il crée la clé de groupe et la distribue aux membres du groupe par l'intermédiaire des contrôleurs locaux. En outre, il effectue le renouvellement périodique et, parfois, occasionnel. Il contrôle toute action concernant la sécurité de groupe effectuée par les contrôleurs locaux.
- Contrôleur local (CL) : délégué par le contrôleur de groupe. Il reçoit la clé de groupe et la distribue aux membres du groupe dans son réseau lors de la configuration initiale du groupe. Un contrôleur local peut jouer le rôle du contrôleur global dans le cas où il y a du changement sur l'état du groupe dans son réseau c'est à dire effectuer le renouvellement occasionnel. De plus, il peut créer et

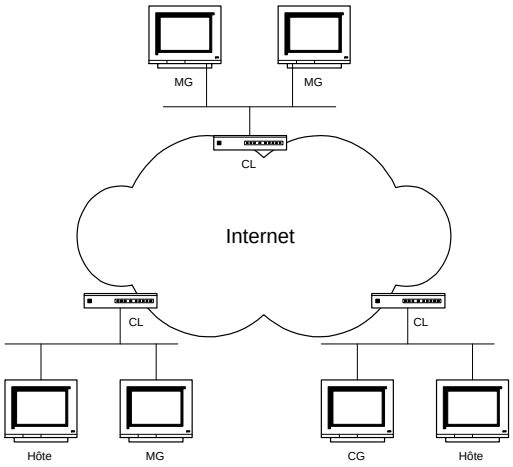


FIG. 4.2 – Architecture de BAAL

distribuer une nouvelle clé de groupe, accepter ou refuser un membre et notifier tout changement dans le groupe aux autres contrôleurs.

- Membre du groupe (MG) : membre de la liste Participant_List ou tout membre qui rejoint le groupe ultérieurement.

Dans ce qui suit, nous allons présenté les différentes opérations requises pour le fonctionnement du protocole :

Initialisation du groupe

Cette étape se propose de distribuer de manière sûre la clé de groupe K_{grp}, à tous les éléments de la liste Participant_List, qui veulent participer au groupe. Pendant cette phase se fait aussi la délégation des contrôleurs locaux pour garantir l'accès au groupe de manière locale et la coopération avec les autres contrôleurs pour gérer la clé de groupe et contrôler l'accès au groupe.

L'initialisation de groupe se découpe en deux phases : la phase d'invitation qui est réservée à l'invitation des membres de la liste Participant_List à participer à la communication de groupe, et la phase de distribution de K_{grp} qui a pour objectif de distribuer de manière sûre la clé de groupe aux membres du groupe et d'authentifier les contrôleurs délégués.
Pendant la phase d'invitation, l'émetteur d'un message inclut dans le message un token signé qui assure l'authentification. Le contrôleur global commence par envoyer un message KC_mg1, à un élément de la Participant_List, contenant la clé publique du CG, son token signé et l'adresse IP multicast du groupe. Le contrôleur local du destinataire du message authentifie l'émetteur ; si l'authentification réussit il stocke la clé publique du CG et ré-envoie le message au destinataire, qui à son tour authentifie l'émetteur ; si l'authentification réussit et s'il accepte de joindre le groupe, il acquitte par un message IGMP-Report contenant son token signé et l'adresse IP multicast du groupe.

En recevant le message IGMP-Report, le CL authentifie l'émetteur. En cas de réussite, il ajoute le nouveau membre à sa liste Local_Participant_List, et envoie au CG un deuxième message KC_mg2 contenant le token signé et la clé publique du nouveau membre, son token signé et sa clé publique.

Le CG va authentifier l'émetteur, en cas de réussite, il l'accepte comme contrôleur local délégué et stocke sa clé publique et l'ajoute à la liste des contrôleurs locaux LCL.

Avant de commencer la phase de distribution de la clé du groupe, le CG construit un paquet, pkt_init contenant deux clés de groupe, K_{grp} et KEK (Key Encryption Key), l'identité du groupe et son identité. Ce paquet sera envoyé en point à point à tous les membres de LCL, dans un message KD_mg1. Ce message est crypté avec la clé publique du destinataire.

En recevant le message KD_mg1, le CL extrait le paquet pkt_init et l'acquitte par un message KD_mg2, contenant l'identité du CL et celle du groupe.

Finalement, le CL envoie le pkt_init à tous les membres de sa liste Local_Participant_List, crypté avec leurs clés publiques respectives.
On note que dans le cas où une infrastructure de clés publiques, PKI (Public Key Infrastructure), est déployée, l'envoi des clés publiques n'est plus obligatoire.
La figure 4.3 illustre la phase d'initialisation de groupe.

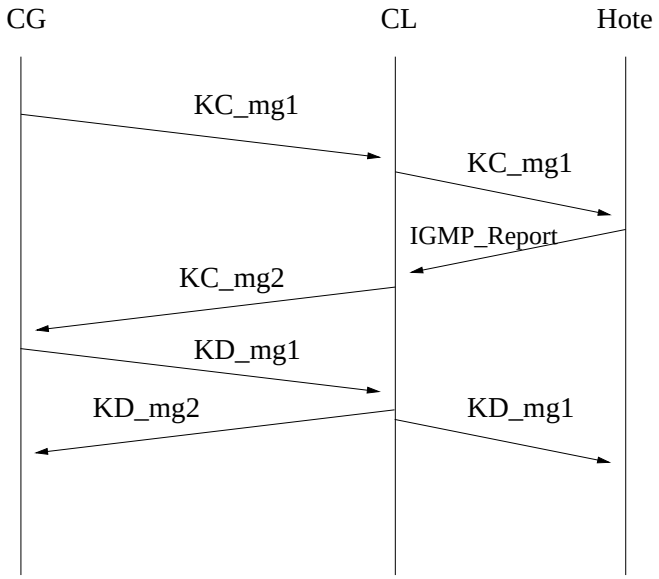


FIG. 4.3 – Initialisation du groupe

Ajout d'une nouvelle entité

Un hôte voulant joindre un groupe multicast selon le protocole BAAL, doit vérifier certaines conditions ; entre autres, il ne doit pas avoir été expulsé du groupe. Il commence par envoyer un IGMP_Report à son contrôleur local, en incluant dans ce message son token signé et l'adresse IP multicast du groupe. Le contrôleur local authentifie le token signé, en cas de réussite, il vérifie que l'entité voulant joindre le groupe n'apparaît pas dans la liste Recovery_List, c'est à dire il n'a pas été expulsé auparavant du groupe. Si le nouveau membre vérifie toutes les conditions pour joindre le groupe, le contrôleur local procède à la phase de renouvellement de la clé de groupe. Ainsi, s'il reçoit un message contenant une nouvelle clé K_{grp}, il ré-envoie cette clé à tous ses membres y compris le nouveau membre. Sinon, deux cas peuvent se présenter :

- Le contrôleur local est délégué, il procède donc à la construction de deux nouvelles clés K_{grp}' et KEK', il construit ensuite un paquet pkt_rekey_join contenant les deux nouvelles clés, l'identité du groupe, son identité et son numéro de priorité. Ensuite, il envoie ce message à tous les membres et contrôleurs du groupe en multicast, crypté avec la clé KEK. Ensuite, il envoie en point à point ce paquet à la nouvelle entité, crypté avec la clé publique du destinataire.
- Le contrôleur local n'est pas encore délégué, il doit alors négocier avec le CG pour obtenir la permission de participer à la gestion de la clé du groupe. Il commence par envoyer un message KC_mg2 au CG, qui à son tour, construit deux nouvelles clés K_{grp}' et KEK', puis il construit un paquet pkt_reKey_join, contenant ces deux nouvelles clés, l'identité du groupe et son identité à lui. Ce paquet sera envoyé en multicast à tous les membres et contrôleurs du groupe, crypté avec la clé KEK, et en point à point à la nouvelle entité joignant le groupe, crypté avec sa clé publique. La figure 4.4 illustre ce cas.

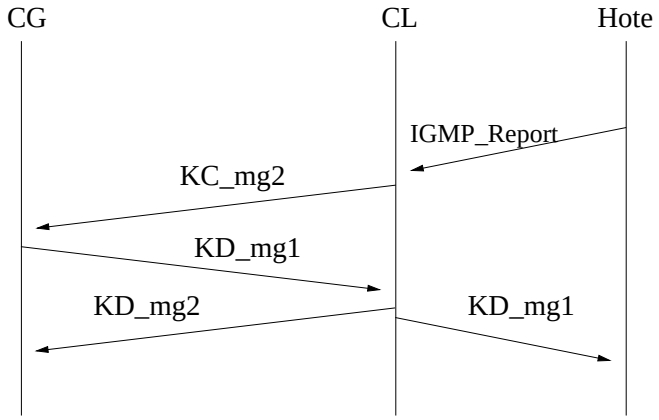


FIG. 4.4 – Ajout d'une nouvelle entité au groupe

Retrait d'une entité

Dans le cas où le retrait est volontaire ou silencieux, le membre voulant quitter le groupe envoie un IGMP_Leave dans le but de stopper le flux de trafic du groupe.

Cependant, au cas où un membre peut mettre la sécurité du groupe en péril, il doit être expulsé et la clé du groupe doit être renouvelée impérativement. Dans ce cas, le CL du membre à expulser crée deux nouvelles clés K_{grp}' et KEK', et construit un paquet pkt_rekey_evict. Ce message de renouvellement occasionnel sera envoyé avec l'identité du groupe, son identité et l'identité du membre à expulser, à tous les membres du groupe, chiffré avec la clé K_{grp}. Comme le membre qui ne fait plus partie du groupe détient encore la clé K_{grp}, il sera capable d'obtenir la nouvelle clé. La solution proposée par BAAL est que le contrôleur local du membre exclu envoie en point à point le message de renouvellement de la clé, à tous ses membres locaux, chiffré avec leurs clés publiques respectives, sauf le membre quittant le groupe.

Renouvellement périodique

Généralement, les clés cryptographiques ont une durée de vie limitée et doivent être périodiquement renouvelées. Ce renouvellement pourra être effectué par le contrôleur global ou par les contrôleurs locaux. Le contrôleur, responsable de ce renouvellement, génère deux nouvelles clés K_{grp}' et KEK' et construit un paquet de renouvellement de clés pkt_rekey_period, contenant en plus des deux nouvelles clés, l'identité du contrôleur, l'adresse IP multicast du groupe et le numéro de priorité du contrôleur. Ce paquet sera crypté avec la clé KEK et envoyé en multicast à tous les membres du groupe.

4.3 Le protocole IOLUS

L'architecture IOLUS est conçue pour remédier au problème 1 affects n et pour assurer la scalabilité en terme de nombres d'abonnés aux groupes multicast. L'arbre de distribution multicast est composé de plusieurs sous groupes arrangés en hiérarchie, formant ainsi une seul groupe multicast virtuel sécurisé.

La figure 4.5 illustre l'architecture IOLUS.

Chaque sous groupe de l'arbre multicast est relativement indépendant, chaque sous groupe a son propre groupe multicast avec sa propre adresse IP multicast, et ce groupe peut être créé avec n'importe quel protocole de routage multicast (PIM, DVMRP,...). Ainsi, chaque sous groupe détient une clé locale appelée K_{SGRP} ; ainsi, quand un membre joint ou quitte le groupe, seulement la clé locale devra être changée.

IOLUS introduit deux types d'agents de sécurité de groupe, GSC (Group Security Controller) contrôlant le sous groupe du premier niveau, et les GSIs (Group Security Intermediaries), un GSI par sous groupe.

Tous les agents de sécurité forment une hiérarchie de sous groupes au sein du groupe multicast virtuel. Le GSC est responsable de la sécurité dans tout le groupe tandis que les GSI sont des entités de

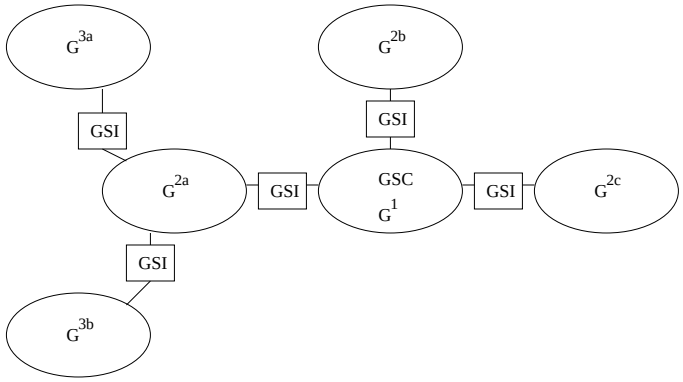


FIG. 4.5 – Exemple d'un arbre multicast IOLUS

confiance autorisées à agir pour contrôler la sécurité dans leurs sous groupes respectifs. Les GSIs sont aussi responsables de recevoir les données multicast depuis leurs GSIs parents ou fils et de les ré-envoyer à leurs GSIs fils ou parents respectivement. Ces GSIs jouent ainsi le rôle de ponts entre les différents niveaux de hiérarchie du groupe en entier.

Pour pouvoir acheminer des données sécurisées entre les différents sous groupes, des opérations de cryptage et de décryptage sont indispensables au sein des différents GSIs.

On note que la hiérarchie imposée aux différents agents de sécurité de groupe a pour rôle de bien montrer la délégation du contrôle entre le GSC et les différents GSIs ; en plus, cette hiérarchie s'avère indispensable lors des opérations de ré-envoi de données multicast entre différents sous groupes, en évitant des loopbacks éventuels.

La figure 4.6 illustre l'architecture IOLUS.

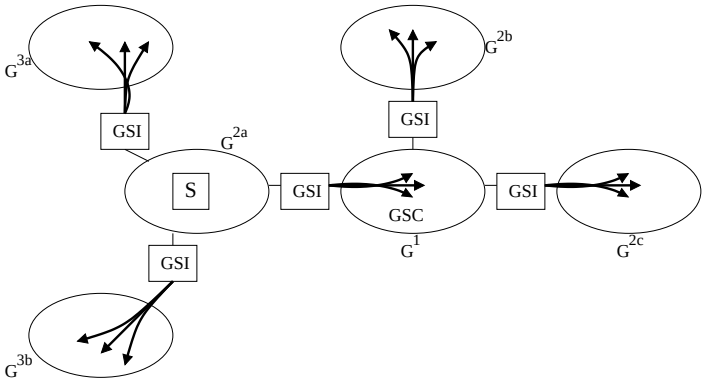


FIG. 4.6 – Transmission de données multicast IOLUS

4.4 Le protocole AMAM

Le protocole AMAM (Active-based Management Architecture for IP Multicast) est un protocole qui intègre différentes tâches de gestion et de sécurité, et distribue les données et les fonctions de gestion. Dans cette section, on va s'intéresser à l'aspect sécurité dans AMAM.

L'architecture se base sur un modèle hiérarchique à trois niveaux : niveau nœud source, niveau nœud intermédiaire et niveau nœud frontal. Elle repose sur la coopération de trois types d'agents de gestion associés à ces niveaux : MSA (Multicast Source Agent), MNA (Multicast Node Agent) et MEA (Multicast Edge Agent). La figure 4.7 illustre cette architecture.

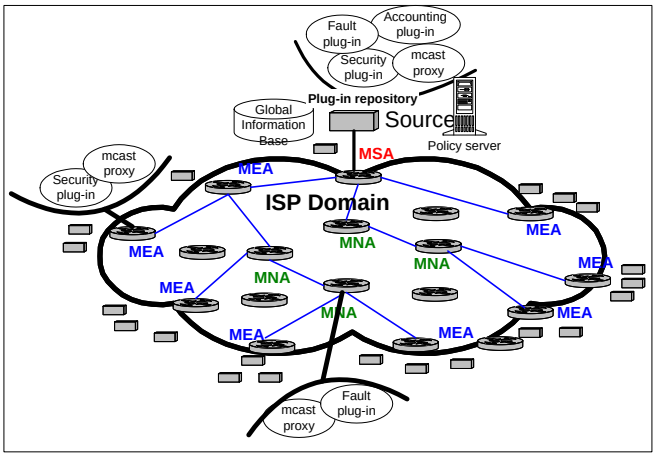


FIG. 4.7 – Architecture globale de gestion de AMAM

Les agents MEA et MNA se déploient et se retirent dynamiquement suivant l'expansion et la contraction de l'arbre multicast. Ces deux agents collectent des informations de gestion pour chaque groupe dans le réseau, et les envoient au MSA qui est le responsable des tâches de gestion.
La sécurité dans AMAM se base sur la génération d'une clé globale par la source et de plusieurs clés locales par les routeurs frontaux. AMAM appartient donc à la deuxième approche de gestion de clé présentée ci-dessus.

Cette architecture s'implémente comme suit :

- L'agent MSA génère une clé globale et la distribue à tous les nœuds de l'arbre multicast. Cette clé est utilisée pour crypter le trafic à délivrer. La clé globale est générée une seule fois mais elle peut être rafraîchie régulièrement pour plus de sécurité.
- Si un routeur possédant la clé globale quitte l'arbre multicast, la clé sera détruite automatiquement.
- Chaque agent MEA génère en local une clé locale et la distribue à tous ses membres.
- L'agent MEA décrypte le trafic avec la clé globale et le re-crypte avec sa clé locale et le diffuse à ses membres.
- Les membres déchiffrent le trafic avec la clé locale.
- A chaque nouvel abonnement/désabonnement enregistré, l'agent MEA regénère une nouvelle clé locale et la distribue à ses membres locaux.

4.5 Le protocole AKMP

Le protocole AKMP (An Adaptative Key Management Protocol for Secure Multicast) est une approche hybride qui se propose de tirer profit des deux approches présentées ci-dessus. L'idée de base de ce protocole pour réaliser la gestion d'un groupe multicast est de commencer avec la première approche tant que la fréquence de changement de la topologie du groupe n'est pas assez grande, et de commuter vers la deuxième approche dès que le taux de dynamicité atteint un seuil donné.

Le protocole commence donc par former un seul groupe de membres qui partagent la même clé. Ce groupe est géré par un routeur AKMP. Durant la session multicast sécurisée, si un routeur AKMP détecte une dynamicité locale dépassant le seuil prédéfini, il forme un sous groupe avec une TEK locale indépendante. Pour ce, il génère et distribue la nouvelle TEK locale à tous les membres de son sous groupe. Cette clé est appelée DK (Downstream Key). Ce routeur entame donc un processus de cryptage/décryptage : il décrypte les données reçues utilisant la clé de son routeur AKMP parent (cette clé s'appelle UK : Upstream Key) et les recrypte utilisant sa DK. On dit aussi que ce routeur est passé d'un état passif à un état actif.

De cette manière, AKMP réduit l'overhead du processus de cryptage/décryptage tout en atténuant le phénomène 1 affects n.

Chaque routeur AKMP R_i détient :

- une fonction d'évaluation de dynamicité f_i : cette fonction mesure le taux de dynamicité du groupe. On peut la calculer de la manière suivante :

  if mcf > d then {switch to dec/rec process}
  f_i = true ;
  else
  f_i = false ;
  end ;
  avec mcf : nombre de changements des membres par unité de temps,
  et d : un seuil prédéfini.

- DK_i et UK_i ; dans le cas où $DK_i = UK_i$, le routeur AKMP est passif.
- une paire de clé (publique K_i et privée K_i^{-1}) permettant ainsi des échanges sécurisés de messages entre les différents routeurs AKMP.

Ainsi l'état d'un routeur AKMP est dynamique, il peut être passif, actif ou en attente d'une nouvelle clé UK :
(1) Actif : un routeur est actif s'il a entamé un processus de cryptage/décryptage ;
(2) Passif : un routeur est passif s'il n'effectue pas des opérations de cryptage/décryptage ;
(3) En attente de UK : un routeur est dans cet état lorsqu'il attend une nouvelle clé d'un routeur AKMP parent qui lui, est actif.

Description du protocole

Quand un routeur AKMP R_i détecte une dynamicité assez forte dans son sous groupe ($f_i = true$), il passe à l'état actif (au cas où il ne l'était pas), il génère ainsi sa DK_i et la distribue à tous ses membres locaux ; il doit ensuite envoyer sa clé ancienne $oldDK_i$ à son routeur AKMP parent pour qu'il change sa clé locale en cas d'égalité avec $oldDK_i$ (ce cas correspond au premier passage du routeur AKMP à l'état actif).

* Cas de Join :

for j :1..nb_attached_members
R_i –> u_j : $\{newDK_i\}_{Kuj}$

for j :1..nb_child_AKMP_routers
R_i –> R_j : $< UPDATE\_UK, \{newDK_i\}_{Kj} >_{K_i^{-1}}$
R_i –> Parent_Router : $< DK\_UPDATED, oldDK_i >_{K_i^{-1}}$

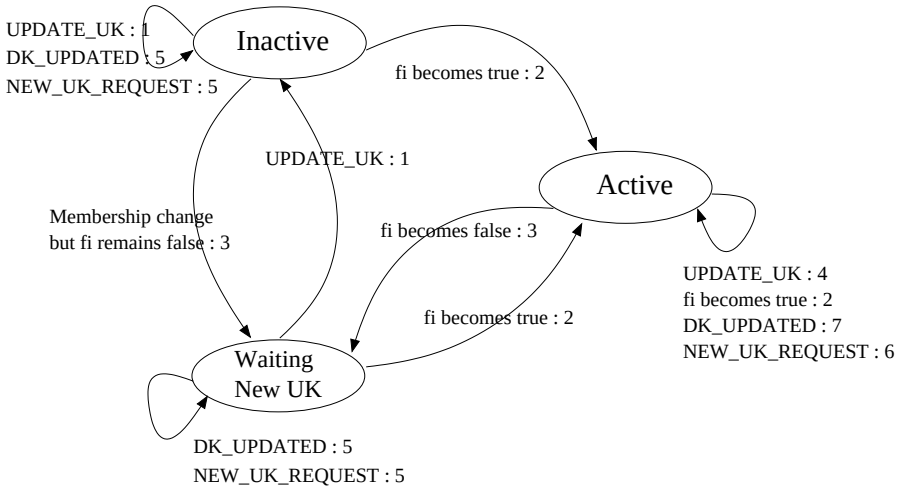
* Cas de Leave : (u_j est le membre quittant le groupe)

for p :1..nb_attached_members, p différent de j
R_i –> u_p : $\{newDK_i\}_{Kp}$

for p :1..nb_child_AKMP_routers
R_i –> R_p : $< UPDATE\_UK, \{newDK_i\}_{Kp} >_{K_i^{-1}}$
R_i –> Parent_Router : $< DK\_UPDATED, oldDK_i >_{K_i^{-1}}$

Au cas où f_i est false, le routeur AKMP reste passif, mais il doit notifier son routeur AKMP père, qui doit renouveler la clé de session du groupe ; ainsi le routeur AKMP en question passe à l'état d'attente de clé.

La figure 4.8 illustre l'automate correspondant à un routeur AKMP.



Legend : Event : Action

1 : DKi = newUK
UKi = newUK
Action 2

2 : Distributes newUK depending on whether the membership change is a join or a leave

3 : sends NEW_UK_REQUEST upstream

4 : UKi = newUK

5 : forwards the message upstream

6 : generates and distributes new DKi

7 : if oldDKj = currentDKi then action 6

FIG. 4.8 – Automate d'un routeur AKMP

4.6 Conclusion

Plusieurs travaux de recherche ont mis en place des architectures de gestion de clés de groupes multicast dans les réseaux filaires. Ces approches peuvent être classées en deux familles. La première consiste à partager une seule clé à tous les membres du groupe, cette approche présente le problème 1 affects n. La deuxième approche consiste à subdiviser le groupe en des sous groupes, chaque sous groupe détient sa clé locale ; cette approche présente donc l'inconvénient d'avoir un overhead de cryptage/décryptage, mais assure plus de "scalabilité". Une approche hybride consiste à adapter l'approche de gestion de clé selon la dynamicité du groupe ; elle permet donc plus de scalabilité tout en atténuant l'overhead de cryptage/décryptage. Dans le chapitre suivant, sera présentée une nouvelle architecture de gestion de clés dans les réseaux Ad Hoc, basée sur l'approche hybride.

Chapitre 5

Nouvelle approche de gestion de clé de groupe dans les réseaux Ad Hoc

5.1 Introduction

Nous avons présenté dans le chapitre précédent différentes approches pour la gestion de clés de groupe multicast dans les réseaux filaires, ces approches sont inadaptées à la nature des réseaux Ad Hoc. En effet, elles ne prennent pas en compte les différentes caractéristiques des réseaux Ad Hoc. Dans ce chapitre, on va présenter une nouvelle approche de gestion de clés dans les réseaux Ad Hoc, basée sur une adaptation de l'approche BAAL présentée dans le chapitre précédent.

5.2 Contexte et Principe

Le contexte de cette approche étant un ensemble de nœuds Ad Hoc, ayant la capacité de router des informations entre eux, en unicast et en multicast. Cette approche se veut générique et indépendante des protocoles de routage utilisés. Ainsi, la phase de gestion de la clé du groupe commencera une fois que l'arbre multicast est construit selon un protocole de routage multicast comme MOLSR ou MAODV.

Le but de cette proposition est de mettre en place un ensemble de mécanismes permettant d'assurer les différents services de sécurité pour les réseaux Ad Hoc à savoir l'authentification, la confidentialité, l'intégrité et la non-répudiation. Ces mécanismes nécessitent une gestion de clé de groupe. La disponibilité n'étant pas assurée par cette approche. On note K_{grp} la clé du groupe partagée entre tous les membres du groupe.

Les opérations définies pour la gestion de la clé du groupe sont les mêmes que celles définies dans BAAL, à savoir, l'initialisation du groupe (distribution de la clé du groupe), l'ajout et le retrait d'une nouvelle entité (renouvellement de la clé du groupe), et le renouvellement périodique de la clé du groupe.

On suppose qu'une infrastructure à clés publiques est mise en place dans notre environnement, ainsi chaque nœud M_i détient une clé privée k_i et une clé publique K_i. Cette infrastructure pourrait être mise en place en utilisant un système PKI (Public Key Infrastructure).

La génération de la clé du groupe se fera en adoptant l'approche de cryptographie à seuil, et la distribution de cette clé sera réalisée selon l'approche BAAL, tout en apportant les modifications nécessaires tenant compte de la dynamicité et de la mobilité des réseaux Ad Hoc. Le principe de fonctionnement de la nouvelle approche est le suivant :

- La gestion de la clé du groupe est, à l'initialisation du groupe, à la charge du contrôleur global (CG), puis cette responsabilité est déléguée à des contrôleurs locaux (CL) selon la dynamicité du groupe. Le contrôleur global est la source de groupe. Les contrôleurs locaux sont des nœuds mobiles, ayant la capacité de détecter la présence des membres locaux et de leur router les données multicast émises par la source. Initialement, tous les membres du groupe, ayant des membres fils, sont considérés des

contrôleurs locaux passifs, et puis selon la dynamicité du groupe, ces nœuds pourront devenir des contrôleurs locaux actifs. La notion de passif et actif est analogue à celle présentée dans AKMP.

- Le contrôleur global commence par calculer une clé du groupe qu'on nomme K_{grp} selon la cryptographie à seuil et en adoptant l'optimisation B-Unicast. Pour cela, si une configuration (n,t+1) est mise en place, le CG consulte sa table de routage et vérifie s'il connaît la route vers t+1 serveurs. Si c'est le cas, il leurs envoie en unicast des messages Key_Query, sinon il diffuse ces demandes dans tout le réseau. A la réception d'un message Key_Query, un serveur commence par authentifier son émetteur. Si l'authentification réussit, il génère une signature partielle et l'inclut dans un message Key_Resp. Ensuite, il envoie ce message au CG, crypté avec sa clé publique. Le CG reste en attente de t+1 signatures partielles valides envoyées par les t+1 premiers serveurs. A leurs arrivées, il les combine et obtient ainsi la clé du groupe. Les serveurs de clés sont choisis, dès le début, selon leurs capacités de calcul et de mémoire. Ces serveurs seront munis d'algorithmes leur permettant de générer aléatoirement des signatures partielles pour les autres nœuds. Le nombre t+1 de serveurs requis pour cette opération sera déterminé grâce aux simulations.
- Cette clé du groupe sera envoyée en unicast à tous les membres du groupe, cryptée avec leurs clés publiques respectives.
- Une fois que tous les membres initiaux détiennent la clé du groupe K_{grp}, la source peut dejà commencer à envoyer les données multicast (initialement, la source attend les acquittements de tous les membres du groupe). Cette approche correspond jusqu'à ce stade à l'approche A présentée dans le chapitre précédent.
- Quand un nœud M_i détecte une forte dynamicité locale (i.e. il a des membres locaux dépassant un seuil fixé), il décide de passer à l'état contrôleur local actif; il obtient ainsi une clé locale K_i^{loc}, via t+1 serveurs selon la cryptographie à seuil et commence ainsi le processus de cryptage/décryptage. Ce contrôleur local procèdera ainsi à décrypter les données multicast envoyées par son nœud parent M_j, utilisant la clé locale de ce nœud parent K_j^{loc} et puis les re-crypter avec sa clé locale K_i^{loc} et les envoyer à ses membres locaux. On dit que le nouveau contrôleur local forme, avec ses membres locaux, un nouveau cluster.
- Si un contrôleur local reçoit une demande de Join, il doit décider du sort de cette requête directement sans aide du contrôleur global, pour ce, il doit authentifier le nouvel abonné et vérifier s'il appartient à la liste des exclus ou pas (voir BAAL). Si l'authentification du nouvel abonné a réussi, le contrôleur local vérifie s'il doit devenir actif (s'il ne l'est pas encore) ou rester passif, et selon son nouvel état, il calcule une nouvelle clé locale et l'envoie à tous ses membres locaux y compris le nouveau, ou bien il demande à son nœud parent de calculer une nouvelle clé et passe donc à l'état d'attente de clé selon AKMP.
- Lors d'un départ d'un membre du groupe, deux cas peuvent se présenter :
  * Le membre décide de quitter le groupe, dans ce cas, selon l'état du contrôleur local : s'il est actif, il y aura renouvellement de la clé locale et s'il est passif, le CL passera en attente d'une nouvelle clé calculée par son nœud parent actif.
  * Le membre a été exclu du groupe par un contrôleur global ou local par ce qu'il met en péril la sécurité du groupe ou par ce qu'on lui fait plus confiance, dans ce cas, ce membre sera ajouté à la liste des exclus, et il y aura renouvellement de la clé exactenment comme le premier cas.

Remarque 1

On pourrait penser à ne pas renouveler la clé du groupe ou la clé locale lors d'un Leave normal, cette idée est utilisée dans BAAL, mais pour assurer plus de sécurité, on préfère recourir au renouvellement de la clé de cryptage à chaque changement de la topologie de l'arbre multicast.

Remarque 2

A chaque fois où il y a renouvellement d'une clé, locale ou globale, ce sont les serveurs mobiles qui les calculent, selon la cryptographie à seuil, et en adoptant l'optimisation B-Unicast.

5.3 Description de l'approche

5.3.1 Architecture

Les acteurs de notre architecture sont un contrôleur global, des contrôleurs locaux et les membres du groupe.

- Contrôleur global : c'est la source du groupe. Initialement, cette entité détient la liste des participants au groupe Participant_List. Le contrôleur global est le responsable de la génération de la clé du groupe, à la phase de l'initialisation ; il assure aussi le renouvellement périodique de la clé du groupe et la gestion de la sécurité au sein du groupe (contrôler les comportements des contrôleurs locaux et des membres du groupe).
- Contrôleur local : est considéré contrôleur local passif, tout nœud mobile, appartenant à l'arbre multicast (membre ou simple participant), et ayant des nœuds fils auquels il achemine le trafic multicast. Chaque contrôleur local détient la liste de ses membres locaux Local_Participant_List, auxquels il doit acheminer le trafic multicast émis par la source. Un Contrôleur local passif, membre du groupe, détient la même clé de cryptage que son nœud parent, il reçoit le trafic multicast, et le route vers ses membres locaux qui détiennent eux aussi la même clé de cryptage. Si le Contrôleur local passif est un participant à l'arbre multicast sans être membre du groupe, il n'aura pas besoin d'avoir la clé de cryptage de son nœud parent.

  Lorsque le taux local de dynamicité atteint un certain seuil, le contrôleur local décide de passer à l'état actif, c'est ainsi qu'il obtient une clé locale et commence le processus de cryptage/décryptage. Chaque contrôleur local actif doit assurer un renouvellement périodique de sa clé locale, en d'autres termes, il joue le même rôle que le contrôleur global, au sein de son sous-groupe.

  Pour décider de son état, chaque contrôleur local détient une fonction d'évaluation de dynamicité, qui peut se calculer de la manière suivante :

  if (mcf $>$ d1 ou mn $>$ d2) then {switch to dec/rec process}
  f_i = true ;
  else
  f_i = false ;
  end ;
  avec :
  mcf : nombre de changements des membres par unité de temps,
  d1 : un seuil de fréquence prédéfini,
  mn : nombre des membres locaux,
  d2 : un seuil de membres prédéfini.

  La différence qui existe entre cette fonction d'évaluation et celle de AKMP, est que dans AKMP, la fonction d'évaluation ne tient en compte que la fréquence de changements des membres, et non pas leurs nombre. Ceci n'est pas adapté à notre contexte de réseau Ad Hoc pour deux raisons. La première étant que tous les membres du groupe jouent aussi le rôle de routeurs et donc peuvent parfaitement être considérés comme des contrôleurs locaux (s'ils ont des membres fils). La deuxième raison est que lors d'un Leave, le contrôleur local actif est obligé de renouveler la clé et de l'envoyer en unicast à tous les membres de son cluster ; le temps nécessaire pour ce renouvellement est ainsi proportionnel au nombre de membres de ce cluster. Ainsi, on devait introduire le critère de nombre des membres locaux dans cette fonction d'évaluation.

  mn est calculé de la façon suivante : un contrôleur local compte le nombre de membres passifs qui lui sont attachés et leurs descendants, et le nombre de membres actifs qui lui sont attachés sans leurs descendants. La figure 5.1 montre un exemple de calcul de mn, d2 est choisi égal à 4.
- Membre du groupe : c'est un membre de la liste Participant_List ou tout membre qui rejoint le groupe ultérieurement.

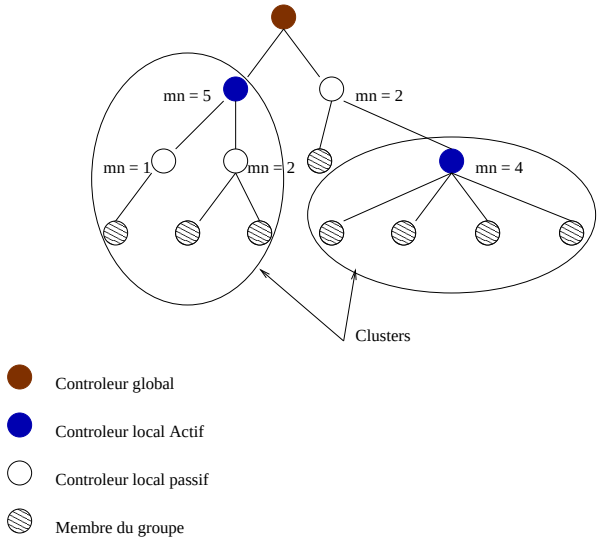


FIG. 5.1 – Exemple d'évaluation du nombre des membres

Remarque 1

Tous les contrôleurs (global et locaux) partagent une même liste appelée Recovery_List, qui contient tous les membres qui ont été exclus du groupe ; cette liste sera utilisée lors d'un Join d'un nouveau membre au groupe.

5.3.2 Initialisation du groupe

A l'initialisation du groupe, le contrôleur global qu'on note CG, initialise les deux listes Participant_List et Recovery_List, Participant_List contiendra tous les membres du groupe appartenant à l'arbre multicast (information divulguée par le contrôleur du protocole de routage multicast). Recovery_List sera initialement vide. Les contrôleurs locaux initialisent à leur tour, leurs listes de membres locaux Local_Participant_List.

Le CG entame ensuite la phase de génération et de distribution de la clé du groupe à tous les membres de la liste Participant_List. La génération de la clé du groupe se fera en utilisant la cryptographie à seuil, comme présenté ci-dessus.

Afin de mieux protéger les messages envoyés lors de la distribution des clés, l'émetteur d'un message y inclut un token signé. Ce token signé forme une partie essentielle du processus d'authentification des messages échangés. Il aide un récepteur à vérifier l'origine du message et l'identité de l'émetteur. Pour définir un token, nous reprenons la forme définie dans [T.B96], où un token contient :

- l'identité unique du récepteur, par exemple son adresse IP ;
- une estampille ;
- un nombre pseudo-aléatoire pour protéger le récepteur contre le rejeu du message.

Le token est inclus dans le message, signé avec la clé privée de l'émetteur.

Le CG envoie le message suivant à tous les membres M_i de la Participant_List crypté avec leurs clés publiques respectives K_i, ce message contient la clé du groupe, l'identité du groupe, l'identité du CG et son token signé.

$$\text{CG} -> M_i : < K_{grp}, IDG, ID_{CG}, [token\_CG]_{Prv\_CG} >_{K_i}$$

avec IDG : identité du groupe, ID_{CG} : identité du CG, $[token\_CG]_{Prv\_CG}$: token signé du CG.

En recevant ce message, chaque membre du groupe le déchiffre, authentifie le CG, extrait la clé du

groupe et l'acquitte avec un message Report [1] contenant l'identité du groupe, l'identité du membre et le token signé du membre, cryptés avec la clé publique du CG.

M_i –> CG : $< IDG, ID\_M_i, [token\_M_i]_{Prv\_M_i} >_{Pbk\_CG}$

avec Pbk_CG : clé publique du CG, $[token\_M_i]_{Prv\_M_i}$: token signé de M_i.

Remarque 2

On a supposé que les membres du groupe connaissent la clé publique du CG, mais on pourrait penser que le CG envoie sa clé publique à tous ses membres lors du premier message.

5.3.3 Ajout d'une nouvelle entité

L'entité désirant rejoindre le groupe envoie un message Report à son contrôleur local, contenant son token signé, son identité et l'identité du groupe.

M_i –> CL : $< IDG, ID\_M_i, [token\_M_i]_{Prv\_M_i} >_{Pbk\_CL}$

avec CL : Contrôleur local de la nouvelle entité.

A la réception de ce message, le contrôleur local authentifie le token signé. Si l'authentification réussit, il donne suite à la demande en vérifiant que l'hôte n'apparaît pas dans la liste Recovery_List, c'est à dire que l'hôte n'a pas été expulsé.

A ce stade , le contrôleur local calcule sa fonction d'évaluation de la dynamicité, deux cas se présentent :

- si $fi = true$, le contrôleur passe à l'état actif, au cas où il était passif. Il doit donc générer une nouvelle clé K_i^{loc} et la distribuer à ses membres locaux : Pour cela, il envoie cette nouvelle clé en multicast à tous les anciens membres locaux, cryptée avec la clé ancienne old_K_i^{loc}, et envoie la même clé K_i^{loc} au nouvel abonné, cryptée avec sa clé publique. En plus, le contrôleur local doit envoyer sa clé locale ancienne à son nœud parent actif M_l pour que ce nœud change sa clé locale pour tous ses membres locaux au cas où old_$K_i^{loc} = K_l^{loc}$ (c'est à dire au cas où le contrôleur local vient de passer à l'état actif, et donc sa clé locale ancienne est égale à la clé de son nœud parent actif).

  for j :1..nb_old_attached_members
  CL –> M_j : $< K_i^{loc} >_{old\_K_i^{loc}}$
  CL –> M_i : $< K_i^{loc} >_{K_i}$ avec M_i : nouvel abonné
  CL –> Active_Parent_Node M_l : $< old\_K_i^{loc} >_{K_l}$ avec K_l : clé publique du nœud parent M_l.

- si $fi = false$, le contrôleur local reste à l'état passif, et donc il doit envoyer une demande de renouvellement de clé à son nœud parent actif, qui entame la génération d'une nouvelle clé et sa distribution à tous ses fils.

Remarque 3

Pour assurer une distribution fiable de la clé du groupe, il suffit que chaque membre acquitte son contrôleur de sa réception de la nouvelle clé. Un délai pourrait être défini, au delà duquel, si le contrôleur ne reçoit pas un acquittement d'une entité, il lui envoie en unicast la nouvelle clé du groupe. Nous supposons dans notre approche que la distribution de la clé se fait de manière fiable.

[1] Dans les réseaux filaires, on utilise IGMP_Report ou MLD_Report, ceci n'est plus possible dans les réseaux Ad Hoc car les messages Report sont dépendants des protocoles de routage utilisés

5.3.4 Retrait d'une entité

Nous distinguons deux cas : retrait volontaire et retrait obligatoire ou expulsion. Le premier cas est un retrait silencieux, qui est réalisé quand un membre veut quitter le groupe et envoie un message Leave incluant son token signé à son contrôleur local dans le but de stopper le flux de trafic multicast du groupe. Dans ce cas, le contrôleur local supprime ce membre de sa liste Local_Participant_List et entame une phase de renouvellement de clé.

Le deuxième cas (expulsion) est connu sous le nom de révocation de membre, et a lieu quand un membre peut mettre la sécurité du groupe en péril, ou peut divulguer la clé du groupe ou simplement son contrôleur local ne lui fait plus confiance. Dans ce cas, le contrôleur local ajoute le nom de ce membre à la liste des exclus Recovery_List, et le supprime de sa liste Local_Participant_List, ensuite il entame une phase de renouvellement de clé.

La phase de renouvellement de clé commence par évaluer la dynamicité locale afin de décider du nouvel état que doit adopter le contrôleur local. Comme dans le cas de l'ajout d'une entité, deux cas se présentent :

- si $fi = true$, le contrôleur passe à l'état actif, au cas où il était passif. Il doit donc générer une nouvelle clé et la distribuer à ses membres locaux. Pour cela, il envoie la nouvelle clé locale K_i^{loc} en unicast à tous les anciens membres locaux, en excluant le membre qui a quitté le groupe, cryptée avec leurs clés publiques respectives. En plus, le contrôleur local doit envoyer sa clé locale ancienne à son nœud parent actif M_l pour que ce nœud change sa clé locale pour tous ses membres locaux au cas où old_$K_i^{loc} = K_l^{loc}$ (c'est à dire au cas où le contrôleur local vient de passer à l'état actif, et donc sa clé locale ancienne est égale à la clé de son nœud parent actif).

  M_s est le membre quittant le groupe,
  for j :1..nb_attached_members, j différent de s,
  CL –> M_j : $< K_i^{loc} >_{K_j}$
  CL –> Active_Parent_Node M_l : $< old\_K_i^{loc} >_{K_l}$ avec K_l : clé publique du nœud parent M_l.

- si $fi = false$, le contrôleur local reste à l'état passif, et donc il doit envoyer une demande de renouvellement de clé à son nœud parent actif, qui entame la génération d'une nouvelle clé et sa distribution à tous ses fils.

Remarque 4

La liste des exclus du groupe Recovery_List doit être diffusée à tous les contrôleurs et les membres du groupe pour éviter qu'un membre exclu par un contrôleur local, rejoint un autre cluster du groupe. Cette diffusion a lieu juste après une révocation d'un membre. Un algorithme permettant le Merge (fusion de plusieurs versions de tables) doit être implémenté dans chaque contrôleur et membre du groupe.

5.3.5 Renouvellement périodique

Généralement, les clés cryptographiques ont une durée de vie limitée et doivent être périodiquement remplacées par des nouvelles clés. La période de renouvellement de clés est déterminée en fonction de la longueur de la clé et de l'algorithme de génération avec lequel la clé a été créée.

Le renouvellement de clés doit être effectué par le contrôleur global et tous les contrôleurs locaux à l'état actif; ce renouvellement se fait en deux étapes : génération d'une nouvelle clé en utilisant la cryptographie à seuil et distribution de cette clé à tous les fils.

Remarque 5

On aurait pu utiliser, pour la phase de génération de clés, la technique Key Agreement et l'adapter à notre contexte de gestion de clés pour les réseaux Ad Hoc, mais on a opté pour la technique de cryptographie à seuil par ce qu'elle présente moins d'overhead en termes de messages envoyés pour la génération des clés.

5.3.6 Traitements de la mobilité

Notre nouvelle approche se propose d'être adaptée à la nature des réseaux Ad Hoc, elle doit donc supporter la mobilité qui est l'une des principales caractéristiques de ces réseaux. Plusieurs scénarios de mobilité peuvent être réalisés ; dans ce qui suit, nous allons énumérer les plus importants et proposer les solutions nécessaires.

Quand un contrôleur local se déplace

Quand un contrôleur local se déplace, tous ses membres locaux ne pourront plus recevoir le trafic multicast du groupe. Pour remédier à ce problème, ces membres doivent immédiatement s'attacher à un autre contrôleur local et pouvoir ainsi continuer à recevoir leur trafic multicast. Reste à voir comment se fait cette transition d'un cluster à un autre et combien de données vont perdre les membres durant cette transition. On peut déjà distinguer deux cas :

* Le contrôleur local notifie avant son déplacement : cette notification pourrait être sous forme d'un message envoyé en multicast à tous les membres locaux, leur informant qu'ils vont devoir s'attacher à un autre contrôleur local. Ces membres locaux vont ainsi diffuser des messages Report, et s'authentifier auprès d'un autre contrôleur local, qui, en cas de succès, va leur distribuer sa clé locale pour pouvoir continuer la réception du trafic multicast du groupe. Cette notification permet donc de gagner du temps pour que les membres locaux se rendent compte du déplacement de leur ancien contrôleur local.
* Le contrôleur local se déplace sans aucune notification : tous ses membres locaux vont se rendre compte après un certain délai, que le chemin vers la source n'est plus assuré par leur contrôleur local, ils vont ainsi essayer de joindre le groupe avec un autre contrôleur local.

Quand un membre du groupe se déplace

Quand un membre se déplace, il risque de perdre sa réception du trafic multicast, s'il sort de la portée de son contrôleur local. Pour ce, il doit prévoir un autre chemin vers la source, en s'attachant à un autre contrôleur local. La solution est donc d'essayer de s'authentifier auprès d'autres contrôleurs locaux, dès le début du déplacement, au risque d'avoir à un instant donné, deux contrôleurs locaux différents.

Remarque 6

Toutes ces solutions dépendent des politiques de sécurité et de qualité de service mises en place dans le réseau, suivant ces politiques, on pourrait se permettre ou non un temps de latence nécessaire à un nœud qui se déplace, pour pouvoir retrouver sa réception du trafic multicast.

5.4 Conclusion

Tout au long de ce chapitre, on a présenté une nouvelle architecture de gestion de clés dans les réseaux Ad Hoc, basée sur BAAL, tout en apportant les modifications nécessaires pour adapter l'approche à la dynamicité et à la mobilité des réseaux Ad Hoc. Dans le chapitre suivant, sera présenté des simulations visant à spécifier les seuils de fréquence et de membres, utilisés dans la fonction d'évaluation de l'état d'un contrôleur local, en utilisant le simulateur NS.

Chapitre 6

Simulations et résultats

6.1 Introduction

Dans ce chapitre, nous allons présenter des simulations et des résultats qu'on a réalisés pour déterminer le seuil de fréquence et de nombre de membres, au delà desquels, un contrôleur local passe à l'état actif, dans le cadre de la nouvelle architecture de gestion de clé, présentée dans le chapitre précédent.

Dan cette simulation, on étudie le comportement de l'approche 1 affects n, en utilisant la cryptographie à seuil (configuration (3,2)), en essayant de mesurer le temps nécessaire pour le renouvellement de la clé de groupe, lors d'un Join ou d'un Leave, et ce par rapport à la fréquence des évènements et par rapport au nombre de membres du groupe. Ayant ces deux seuils, on pourrait simuler la nouvelle approche et étudier ses performances.

6.2 Modèle de simulation

Pour notre simulation, nous avons utilisé le simulateur NS, version ns2.1b9a [C.L03][ns203]. Le réseau simulé est un réseau Ad Hoc, composé de 100 nœuds, et utilisant MAODV comme protocole de routage multiast. Les mouvements des nœuds sont générés aléatoirement pour pouvoir tenir en compte le facteur de la mobilité. Pour générer des sessions réelles multicast, on utilise le modèle présenté par Almeroth [K.A96], ce modèle a été utilisé aussi pour simuler AKMP dans [H.B02], et suggère que l'arrivée des membres suit un processus poissonien de paramètre $\lambda = 10$, et que la durée d'appartenance à un groupe suit une distribution exponentielle de paramètre $\mu = 145$. Ce modèle a été déduit à partir de réelles sessions multicast observées au sein du Mbone. La figure 6.1 montre la variation du nombre de membre du groupe en fonction du temps, pour le modèle simulé.

6.3 Résultats de la simulation

Avant de commencer la simulation, nous avons dû calculer le coût en terme du temps, d'un Join ou d'un Leave au sein du groupe multicast.

Lors d'un Join, le nombre de message émis pour renouveler la clé est égal à 9, qui se répartissent en :

- 1 message de demande de Join.
- 3 messages qu'envoie le contrôleur global à trois serveurs pour la génération de la nouvelle clé du groupe, et 3 messages de réponse (on suppose qu'il n'y a pas de serveurs compromis).
- 1 message qu'envoie le contrôleur global en multicast à tous les membres du groupe, contenant la nouvelle clé du groupe.
- 1 message qu'envoie le contrôleur global en unicast au nouvel abonné, contenant la nouvelle clé du groupe.

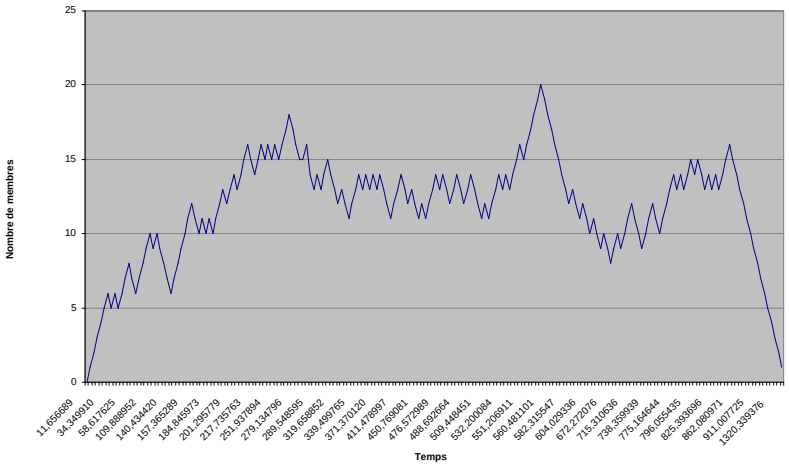


FIG. 6.1 – Variation du nombre de membres du groupe

L'envoi de la clé du groupe nécessite des opérations de chiffrement/déchiffrement, nous avons opté pour l'algorithme 3DES. 8 opérations de cryptage/décryptage seront nécessaires lors d'un join, ces opérations sont les suivantes :

- 3 opérations de cryptage réalisées par les serveurs générateurs de la nouvelle clé.
- 3 opérations de décryptage réalisées par le contrôleur global lors de la réception des 3 clés partielles.
- une opération de cryptage de la nouvelle clé lors de son émission en multicast vers tous les membres du groupe.
- une opération de décryptage de la nouvelle clé, réalisée par le nouvel abonné (on suppose que les autres membres du groupe vont décrypter la clé en parallèle) et donc on ne comptera qu'une seule opération de décryptage.

Le temps moyen d'envoyer un message dans notre réseau est $t = 0.002579$, et le temps de réaliser une opération de cryptage ou de décryptage selon 3DES est $d = 0.000712s$ (crypter ou décrypter un message de 1500 octets). Ainsi nous pouvons estimer le coût moyen d'attente pour le renouvellement de la clé, lors d'un Join à $9 * t + 8 * d = 0.028907s$. Ce coût doit être ajouté au temps de latence nécessaire entre la demande de Join et le Join effectif.

La même étude est faite lors d'un Leave : le nombre de messages émis pour renouveler la clé est égal à (7+n) (avec n : nombre de membres du groupe), qui se répartissent en :

- 1 message de demande de Leave.
- 3 messages qu'envoie le contrôleur global à trois serveurs pour la génération de la nouvelle clé du groupe, et trois messages de réponse (on suppose qu'il n'y a pas de serveurs compromis).
- n messages qu'envoie le contrôleur global en unicast à tous les membres du groupe, contenant la nouvelle clé du groupe.

Pour assurer la confidentialité de la clé du groupe lors d'un leave, on a besoin de (6+2n) opérations de cryptage/décryptage :

- 3 opérations de cryptage réalisées par les serveurs générateurs de la nouvelle clé.
- 3 opérations de décryptage réalisées par le contrôleur global lors de la réception des 3 clés partielles.
- n opérations de cryptage de la nouvelle clé lors de son émission en unicast vers tous les membres du groupe.

- n opérations de décryptage de la nouvelle clé, réalisées par tous les membres de groupes en utilisant leurs clés privées.

La figure 6.2 présente le temps de renouvellement de la clé suite à un Join par rapport au nombre de membres de groupe. Vu que le coût engendré par le renouvellement de la clé est constant, les différentes variations qu'on remarque sur la courbe sont dûes à ce que parfois le temps de latence entre une demande de Join et le Join effectif peut varier selon maodv et selon la disposition du nouvel abonné par rapport aux autres membres du groupe. Cette courbe ne nous permet pas donc de définir un seuil relatif au nombre de membres du groupe.

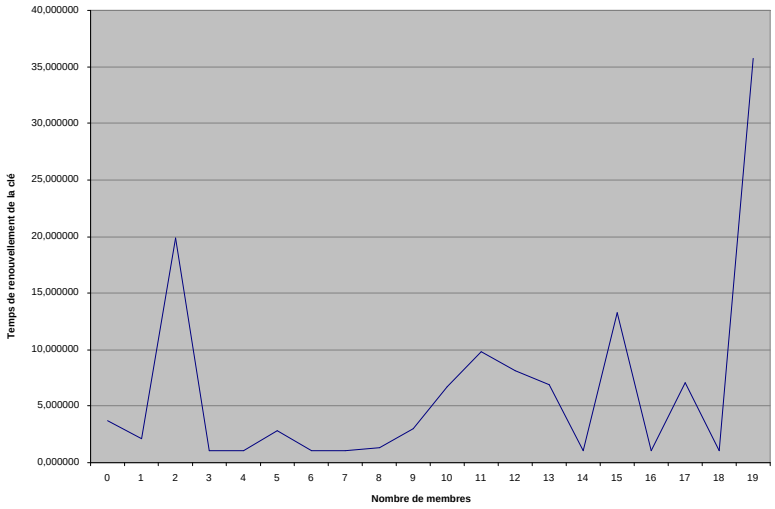


FIG. 6.2 – Temps de renouvellement de la clé suite à un Join par rapport au nombre de membres de groupe

La figure 6.3 montre que le temps de renouvellement de la clé suite à un Leave est proportionnel au nombre de membres de groupe. Ceci nous permet de fixer un seuil de nombre de membres pour former un cluster. Si on prend comme contrainte que le temps de renouvellement de la clé suite à un Leave ne doit pas dépasser 0.05s, le seuil sera alors 8 membres par cluster.

Pour réaliser la figure 6.4, on a calculé le temps de renouvellement moyen de la clé du groupe lors d'un Join ou un Leave, par rapport à des fréquences d'évènements calculées sur des intervalles de temps égaux. Si on prend comme deuxième contrainte que le temps de renouvellement de la clé par rapport à la fréquence ne doit pas dépasser 1s, la figure nous montre que, une fois le régime permanent établi, définir le seuil de fréquence à 9 évènements par unité de temps pourra satisfaire cette contrainte.

6.4 Conclusion

Cette simulation nous a permis de déterminer les deux seuils de fréquence et de nombre de membres par cluster. Le travail restant concernant la simulation consiste à valider ces deux valeurs avec d'autres simulations en essayant de modifier le taille du groupe, le protocole de routage multicast utilisé,... Une fois les deux seuils seront validés, on pourrait simuler la nouvelle approche de gestion de clés, en étudiant ces performances par rapport aux deux approches A et B (voir chapitre 4).

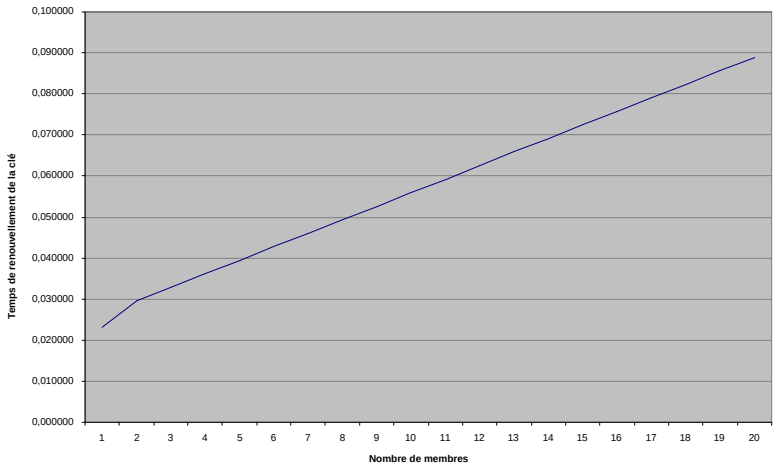


FIG. 6.3 – Temps de renouvellement de la clé suite à un Leave par rapport au nombre de membres de groupe

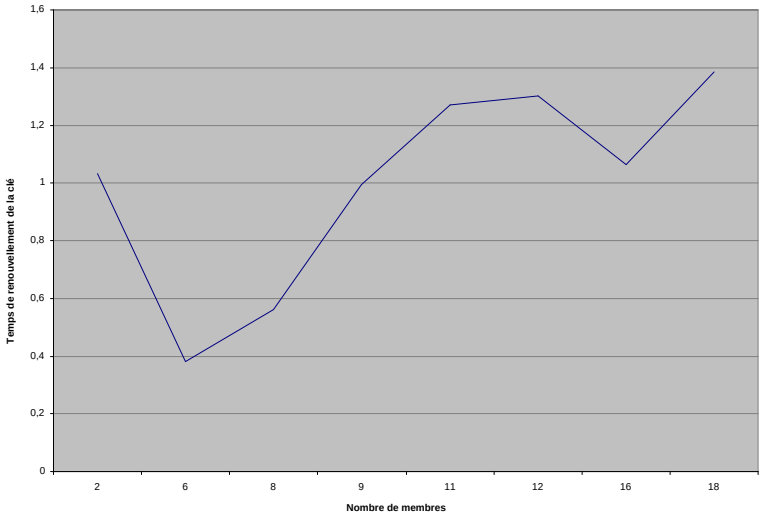


FIG. 6.4 – Temps de renouvellement de la clé par rapport à la fréquence d'évènements

Chapitre 7

Conclusion Générale

7.1 Conclusion

Sécuriser un réseau Ad Hoc est un véritable défi. En effet, ce type de réseaux présente plus de vulnérabilité en terme de sécurité, à cause de ses caractéristiques (absence d'infrastructure, topologie dynamique, bande passante limitée, liens sans fil,...). Pour cela, il faut instaurer au sein d'un réseau Ad Hoc, les services indispensables à la sécurité, qui sont, l'authentification, la confidentialité, l'intégrité, la non-répudiation et la disponibilité.

Parallèlement, le problème de la sécurité a fait surface avec l'arrivée des applications multicast comme les conférences virtuelles sur Internet et le télé-enseignement. Les applications multicast présentent plus de vulnérabilités en terme de sécurité, dans les réseaux Ad Hoc. Une solution appropriée s'avère donc indispensable. Cette solution doit assurer la confidentialité des données multicast transmises au sein d'un groupe, l'authentification de la source et de tous les membres du groupe,... Une approche de gestion de clé au sein du groupe multicast dans un réseau Ad Hoc, pourrait assurer ces différents services de sécurité. Cette approche fera partager une clé de groupe, à tous ses membres, de sorte que, toutes les données multicast transmises depuis la source vers les destinations, seront chiffrées et déchiffrées respectivement, utilisant cette même clé. Le renouvellement de la clé du groupe s'avère indispensable lors d'un changement dans la topologie du groupe ; ainsi, à chaque Join ou Leave au sein du groupe multicast, la clé du groupe doit être renouvelée.

Afin d'instaurer une approche de gestion de clés dans les réseaux Ad Hoc, nous avons étudié le protocole BAAL : ce protocole définit une solution au problème d'extensibilité de la sécurité des communications de groupes dynamiques, dans les réseaux filaires, par le moyen d'une gestion decentralisée de la clé du groupe. BAAL tel qu'il a été réalisé, n'est pas adapté à la nature des réseaux Ad Hoc. Des modifications ont été donc apportées à ce protocole et ont aboutit à une nouvelle architecture de gestion de clé, plus appropriée aux réseaux Ad Hoc.

La nouvelle approche, ainsi élaborée, permet plus de passage à l'échelle, plus de dynamicité et est plus robuste contre les attaques malicieuses qui menacent de plus en plus les réseaux Ad Hoc.

7.2 Perspectives

Comme perspectives de ce travail, on pourrait envisager une validation de la nouvelle approche et l'étude de ses performances, par rapport à d'autres approches de gestion de clés. On pourrait envisager aussi d'augmenter la dynamicité et la scalabilité de la nouvelle architecture, en améliorant la fonction d'évaluation décrite dans le chapitre 5.

La disponibilité n'étant pas assurée par cette approche, on pourrait songer à l'intégrer et ainsi former une nouvelle architecture de sécurisation des communications multicast dans les réseaux Ad Hoc.

Bibliographie

[A.L01] A.Laouiti, P.Jacquet, P.Minet, L.Viennot, T.Clausen et C.Adjih. *Multicast Optimized Link State Routing*, April 2001.

[A.L02] A.Laouiti, P.Jacquet, P.Minet, L.Viennot, T.Clausen, P.Muhlethaler et A.Qayyum. *Optimized Link State Routing Protocol*, November 2002.

[A.L03] A.Laouiti, P.Jacquet, P.Minet, L.Viennot, T.Clausen et C.Adjih. *Multicast Optimized Link State Routing. Report Research 4721*, INRIA, February 2003.

[Chr02] M. E. Christman. *Extensions for Multicast in Mobile Ad Hoc Networks, (XMMAN) :The reduction of Data Overhead in Wireless Multicast Trees.* Thèse de maître, Faculty of the Virginia Polytechnic Institute and State University, July 2002.

[C.L03] C.Lindemann. *http ://rul-www.cs.uni-dortmund.de/MobileP2P/mainE.html*, 2003.

[E.P01] C. E.Perkins. *Ad Hoc Networking.* Addison Wesley, 2001.

[E.R99] E.Royer et C.Perkins. *Multicast Operation of the Ad-Hoc On-Demand Distance Vector Routing Protocol.* Dans *International Conference on Mobile Computing and Networking (MobiCom99)*, numéro 5. Seatle, WA, USA, august 1999.

[F.S99] F.Stajano et R.Anderson. *The Resurrecting Duckling Security Issues for Ad Hoc Wireless Networks. Software Symposium*, September 1999.

[G.C02] G.Chaddoud. *Sécurisation de communication de groupes dynamiques.* Thèse de doctorat, Université Henri Poincaré - Nancy1, Département de formation doctorale en informatique UFR STMIA, Ecole doctorale IAEM Lorraine, Nancy, Aout 2002.

[H.B02] H.Bettahar, A.Bouabdallah et Y.Challal. *An Adaptive Key Management Protocol for secure multicast. IEEE Network*, October 2002.

[H.L] H.Luo, P.Zerfos, Jiejun.Kong, S.Lu et L.Zhang. *Self-securing Ad Hoc Wireless Networks.*

[H.S02] H.Schauer. *Sécurité des réseaux sans fil 802.11B.* HSC Hervé Schauer Consultants, July 2002.

[H.S03] H.Sallay, A.Lahmadi, O.Festor et I.Chrisment. *Extension de l'architecture active AMAM pour le support des services de sécurité multicast.* Dans *GRES'2003 Colloque Francophone sur la Gestion de Réseaux et des Services.* February 2003.

[Hu02] Y. Hu, A.Perrig et D.Johnson. *Packet Leashes : A defense against Wormhole Attacks in Wireless Ad Hoc Networks. Report Research TR01 - 384*, Rice University Department of Computer Science, September 2002.

[J.H] J.Hubaux. *Security of Wireless Ad Hoc Networks.*

[J.J01] J.Jetcheva et D.Johnson. *Adaptive Demand-Driven Multicast Routing in MultiHop Wireless Ad Hoc Networks.* Dans *ASM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001).* October 2001.

[J.L02] J.Leneutre. *Authentification dans les réseuax Ad Hoc : Problématique et état de l'art.* Dans *SAR'2002.* Juillet 2002.

[K.A96] K.Almeroth et M.Ammar. *Collecting and modelling the join-leave behaviour of multicast group members in the Mbone.* Dans *The Symposium on High Performance Distributed Computing.* Syracuse NY, 1996.

[L.C03] L.Costa, M.Amorium et S.Flida. *Utilisant l'inondation controlée pour la découverte de routes dans les réseaux Ad Hoc*. Dans *GRES'2003 Colloque Francophone sur la Gestion de Réseaux et des Services*. February 2003.

[Lee99] S. Lee, C.Chiang et M.Gerla. *On-Demand Multicast Routing Protocol*. Dans *IEEE WCNC'99*. New Orleans, September 1999.

[L.M02] L.Mguyen, R.Safavi-Naini, W.Susilo et T.Wysocki. *Secure Authorization, Access Control and Data Integrity in BlueTooth*. Dans *IEEE International Conference on Networks*, numéro 10. IEEE Singapore Computer Chapter, School of EEE, Singapore Polytechnic, August 2002.

[L.Z99] L.Zhou et Z. J.Haas. *Securing Ad Hoc Networks. IEEE Network*, November/December 1999.

[N.A00] N.Asokan et P.Ginzboorg. *Key Agreement in Ad Hoc Networks*. February 2000.

[ns203] *http ://www.isi.edu/nsnam/dist/*, 2003.

[P.U03] P.Urien et Guy.Pujolle. *Architecture sécurisée par carte à puce pour des réseaux sans fil sures et économiquement viables*. Dans *GRES'2003 Colloque Francophone sur la Gestion de Réseaux et des Services*. February 2003.

[S.C99] S.Corson et J.Macker. *Mobile Ad Hoc Networking (MANET) :Routing Protocol Performance Issues and Evaluation Considerations*, January 1999.

[S.L] S.Levijoki. *Authentification, Authorization and Accounting in Ad Hoc Networks.*

[S.M] S.Mittra. *Iolus : A Framework for Scalable Secure Multicasting.*

[S.Y02] S.Yi et R.Kravets. *Key Management for Heterogeneous Ad Hoc Wireless Networks*, July 2002.

[T.B96] T.Ballardie. *Scalable Multicast Key Distribution*, Mai 1996.

[T.h00] T.hardjono et G.Tsudik. *IP Multicast Security : Issues and Directions. Annales des télécommunications éditée par les groupes des Ecoles des Télécommunications GET et HERMES*, pages 324–340, Aout 2000.

[T.Ka] T.Kunz et E.Cheng. *Multicasting in Ad Hoc Networks : Comparing MAODV and ODMRP.*

[T.Kb] T.Kwon. *Authentification and Key Agreement via Memorable Password.*

[T.L00] T.Lamlouma. *Le routage dans les réseaux mobiles Ad Hoc*, September 2000.

[V.D00] V.Devarapalli, A.Selcuk et D.Sidhu. *MZR : A Multicast Protocol for Mobile Ad Hoc Networks*, November 2000.

[V.L] V.Legrand, F.Abdesselam et S.Ubéda. *Etablissement de la confiance et réseaux Ad Hoc - Un état de l'art.*

[V.V02] V.Varadharajan. *Security for Cluster Based Ad Hoc Networks*, August 2002.

[YD03] Y.Ghamri-Doudane, A.Munaretto et N.Agoulmine. *Une nouvelle architecture de gestion par politiques pour les réseaux ad hoc d'entreprises nomades*. Dans *GRES'2003 Colloque Francophone sur la Gestion de Réseaux et des Services*. February 2003.

ANNEXE

1 Generic Protocol for Password Authenticated Key Exchange

1.1 Version Two-Party [N.A00]

Les deux entités A et B partagent un secret P. Le nœud A détient deux clés E_A et E_B pour crypter et décrypter respectivement, et génère deux chaînes de caractères $Challenge_A$ et S_A ; le nœud B génère trois chaînes de caractères qui sont R, $Challenge_B$ et S_B.

Deux fonctions sont aussi connues de la part de A et B : h(.) qui est une fonction publique et f(.,.) qui est une One-Way fonction. Le but de ce protocole est que les deux nœuds A et B peuvent s'authentifier mutuellement en utilisant le mot de passe faible P, et se mettre d'accord sur la clé de session K qui est égale à f(S_A,S_B).

L'échange de messages se fait comme suit :
(1) A –> B : A, P(E_A)
A envoie à B un label l'identifiant et E_A crypté avec P.
(2) B –> A : P(E_A(R))
B décrypte E_A et envoie à A R crypté avec E_A plus le mot de passe P.
(3) A –> B : R($Challenge_A$,S_A)
A décrypte R, crypte $Challenge_A$ et E_A utilisant R comme clé et envoie le tout à B.
(4) B –> A : R(h($Challenge_A$),$Challenge_B$,S_B)
B décrypte $Challenge_A$, calcule h($Challenge_A$), le crypte avec $Challenge_B$ et S_B en utilisant R comme clé, et envoie le résultat à A.
(5) A –> B : R(h($Challenge_B$))
A décrypte les trois entités et vérifie que la première est bien h($Challenge_A$). Il pourra ainsi vérifier que B a pu recevoir $Challenge_A$. Puis A calcule h($Challenge_B$), le crypte en utilisant R comme clé et renvoie le résultat à B.
La clé de session K sera égale à f(S_A,S_B).

1.2 Version Multi Party [N.A00]

Considérons n participants M_i,i=1,...,n, avec M_n comme leader. M_n détient une paire de clés (E,D) pour crypter et décrypter respectivement. Soit S_i la contribution de l'entité M_i pour la génération de la clé de session K. L'échange de messages se déroule comme suit :
(1) M_n –> ALL : M_n, P(E)
(2) M_i –> M_n : M_i, P(E(R_i, S_i)), i=1,...,n-1
(3) M_n –> M_i : R_i(S_j, j=1,...,n), i=1,...,n-1
(4) M_i –>M_n : M_i, K(S_i, H(S_1, S_2,..., S_n)) pour i donné
On note que la condition de de contribution est très contraignante du moment où M_n doit attendre à ce que tous les M_i, i=1,...,n-1, lui envoient leurs contributions pour qu'il puisse entamer avec la troisième étape.

2 Password Authenticated Diffie-Helmann Key Exchange

2.1 Version Two-Party [N.A00]

Les deux entités A et B se mettent d'accord sur un mot de passe faible P et un générateur g du groupe multiplicatif Z_p^*. A et B choisissent aléatoirement deux secrets S_A et S_B, tels que 1<= S_A, S_B <= P-1.

Les messages entre les deux entités se déroulent comme suit :

(1) A –> B : A, P(g^{S_A})

A calcule g^{S_A}, le crypte avec P et envoie le résultat à B.

(2) B –> A : P(g^{S_B}), K(C_B)

B extrait g^{S_A} du message, calcule g^{S_B} puis calcule la clé de session comme K = $g^{S_A S_B}$. Ensuite, B choisit aléatoirement un Challenge C_B, le crypte en utilisant comme clé K, il crypte aussi g^{S_B} avec la clé P et envoie le tout à A.

(3) A –> B : K(C_A, C_B)

A peut à ce niveau extraire g^{S_B}, il calcule ainsi K. Il exrait aussi C_B qu'il crypte en utilisant comme clé K, avec son challenge choisi aléatoirement C_A, et envoie le résultat à B.

(4) B –> A : K(C_A)

B décrypte le message envoyé par A et peut ainsi vérifier qu'il s'agit de son challenge C_B. Ceci pourrait le convaincre que A connaît le secret faible P. De la même manière, B crypte C_A avec K, et l'envoie à A, qui, en le décryptant, peut vérifier que B connaît à son tour P.

2.2 Version Multi Party [N.A00]

Le contexte de ce scénario est n participants M_1, M_2,..., M_n, partageant un secret faible P. Chaque participant M_i génère aléatoirement une quantité S_i. Le but de ce protocole est de parvenir à une clé de session K égale à $g^{S_1 S_2 ... S_n}$.

Les échanges de messages sont détaillés ci-dessous :

(1) M_i –> M_{i+1} : $g^{S_1 S_2 ... S_i}$, i=1,...,n-2, en séquence

Cette étape nécéssite n-2 envois de messages ; à la fin de cette étape, M_{n-1} aura recu $S_1, S_2, ..., S_{n-1}$ et pourra ainsi calculer $\pi = g^{S_1 S_2 ... S_{n-1}}$.

(2) M_{n-1} –> ALL : $\pi = g^{S_1 S_2 ... S_{n-1}}$, en diffusion

(3) M_i –> M_n : P(C_i), $i = 1, ..., n-1$, en parallèle, avec $C_i = \pi^{\hat{S}_i / S_i}$ et $\hat{S}_i$ est un facteur d'aveuglement choisi aléatoirement par M_i

Chaque M_i enlève de π sa contribution S_i et ajoute un facteur d'aveuglement $\hat{S}_i$.

(4) M_n –> M_i : $(C_i)^{S_n}$, $i = 1, ..., n-1$, en parallèle

M_n décrypte (C_i), lui ajoute son S_n et l'envoie au M_i correspondant.

(5) M_i –> ALL : M_i, K(M_i, H(M_1, M_2,...,M_n)), pour i donné, en diffusion

A ce stage, tous les participants pourront calculer $K = g^{S_1 S_2 ... S_n}$. Le cinquième message est envoyé pour une simple vérification.